Universität Bremen

Fachbereich 3: Mathematik und Informatik

Deutsches Forschungszentrum für Künstliche Intelligenz

Inertial Motion Capturing

Rigid Body Pose and Posture Estimation with Inertial Sensors

Dissertation
zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)

Felix Wenk

Felix Wenk

Inertial Motion Capturing

Rigid Body Pose and Posture Estimation with Inertial Sensors

Dissertation, Fachbereich 3: Mathematik und Informatik

Universität Bremen, 27. September 2016

1. Gutachter: Prof. Dr. Udo Frese
2. Gutachter: Prof. Dr. Michael Lawo

Datum des Kolloquiums: 3. Februar 2017

Zusammenfassung

Diese Dissertation beschäftigt sich mit der Schätzung von Posen, also Positionen und Orientierungen, aus Inertialsensordaten. Betrachtet werden sowohl die Posen einzelner Starrkörper als auch die Posen der Körper eines Skeletts, also eines Systems aus Starrkörpern.

Die wesentliche Einsicht bezüglich der Orientierungsschätzung einzelner Starrkörper ist, sie als Fusion von Inertialsensordaten und ihrem Dynamikmodell mit Vorwissen zu betrachten. Dazu werden verschiedene Kalman-Filter betrachtet, die dieselben Daten mit demselben Dynamikmodell und verschiedenen Annahmen kombinieren. Es stellt sich heraus, dass das klassische Orientierungsschätzer-Messmodell, die Abweichung der Beschleunigungsmessung von der (umgedrehten) Gravitation zu betrachten, äquivalent zu der Annahme ist, dass der Körper im langfristigen Mittel nicht beschleunigt. Die dann naheliegenden Annahmen, dass im langfristigen Mittel die Geschwindigkeit null ist, oder dass der Körper im langfristigen Mittel an derselben Position bleibt, führen ebenfalls zu Orientierungsschätzern.

Der Orientierungsschätzer, der sich aus der Positionsannahme ergibt, schätzt darüber hinaus die Position des Körpers, die lokal akkurat ist; sie folgt über einen kurzen Zeitraum den Beschleunigungssensordaten. Die Position driftet aber nicht unbegrenzt weg, was bei der ausschließlichen Integration der Sensordaten entsprechend ihres Dynamikmodells der Fall wäre. Der Fokus bei der Orientierungsschätzung für einzelne Starrkörper liegt auf dem Zusammenspiel zwischen Dynamikmodell und Vorwissen. Beispielsweise enthält die integrierte Position keine verwertbare Information, wenn man annimmt, dass die Geschwindigkeit im langfristigen Mittel null ist.

Im zweiten Teil der Arbeit werden die Posen aller Körper eines Skeletts, also dessen Haltung, ausschließlich aus Inertialsensordaten geschätzt. Insbesondere werden keine Magnetometer verwendet, um die Drehung um die Vertikale zu bestimmen.

Werden die Inertialsensordaten aller Körper eines Skeletts mit dem Vorwissen, dass die Körper über Gelenke miteinander verbunden sind, fusioniert, werden die relativen Orientierungen der einzelnen Körper untereinander beobachtbar, die Relativdrehungen um die Vertikale eingeschlossen: Messen zwei Beschleunigungssensoren auf über ein Gelenk verbundenen Körpern die Beschleunigung einer Bewegung in unterschiedlichen Richtungen, lässt sich daraus die relative Orientierung der Körper ableiten, wenn die Effekte der Bewegung des Gelenks herausgerechnet werden. Die Drehung des Skeletts insgesamt um die Vertikale lässt sich so natürlich noch immer nicht bestimmen, ist für die Körperhaltung meist aber auch nicht relevant.

Der vor diesem Hintergrund entwickelte Haltungsschätzer wird im Sensoranzug SIRKA zum Einsatz gebracht. Um auf der sehr eingeschränkten Hardware des Anzugs in Echtzeit Körperhaltungen schätzen zu können, werden die Sensordaten vorverarbeitet, sodass Schätz- und Samplingrate der Sensoren voneinander unabhängig werden, ohne viel Information zu verlieren. Darüber hinaus muss die Platzierung der Sensoren innerhalb der Kleidung nicht justiert werden, sondern wird automatisch kalibriert. Die so entwickelte Motion-Capturing-Arbeitskleidung wird eingesetzt, um auf einer Werft während Schweißarbeiten die Körperhaltung des Schweißers zu schätzen. Mit einem Motion-Capturing-Anzug, der sich auf Magnetometer verlässt, wäre das nicht möglich.

Abstract

This dissertation is about estimating poses from inertial sensor data, that is estimating orientations and positions. Both poses of single rigid bodies as well as poses of so called skeletons, i. e. systems of jointed rigid bodies, are covered.

The key insight into orientation estimation of a single rigid body is to view it as the fusion of sensor data and its dynamics model with prior information. To this end, three different Kalman Filter variations are presented, which fuse the same sensor data and the same dynamics with three different priors. It turns out that the classical model to correct the inclination in an orientation estimator, namely comparing the accelerometer measurement with (negative) gravity, is equivalent to the assumption that the rigid body does not accelerate on long-term average. Assuming that the velocity is zero on long-term average or that the rigid body stays at the same position on long-term average are alternative assumptions and both priors also yield orientation estimators.

Moreover, the orientation estimator resulting from the position assumption also estimates a position, which is locally accurate — it follows the accelerometer measurements — but does not drift unboundedly, which it would if the position were obtained by integrating according to the dynamic model only. The focus here is more on the interplay of inertial sensor data and its dynamic model with prior information than it is on practical applications. For instance, for the integrated position to be a usable quantity, the estimate has to be conditioned on the long-term average of the position being zero instead of the velocity or acceleration being zero.

In the second, bigger part of this dissertation the posture of a skeleton, i. e. the poses of all the skeleton's bodies, are estimated, again using inertial sensor data only. Notably, no magnetometers are used to recover the rotations around the vertical. Without magnetometers, the rotation of the skeleton as a whole around the vertical, of course, can not be estimated. However, to asses the skeleton's posture, it is also not important.

If inertial sensor data of all bodies is fused with the prior information that a skeleton's bodies are jointed using hinges and spherical joints, the relative orientations of the bodies become observable completely: If two accelerometers of two jointed bodies measure the acceleration of a motion, then the relative orientation of those two bodies can be recovered from the directions of the accelerometer measurements, if effects due to movements of the joints are compensated for.

The posture estimator that exploits this insight is developed and used in the sensor suit SIRKA, which is workwear with inertial sensors embedded into the clothing. On computationally very limited hardware, which is completely integrated into the suit, the estimator yields posture estimates in real-time. To make this possible, a technique to decouple the sensor's sampling rate from the estimation rate is introduced. Moreover, the sensor orientations and positions inside the suit are almost arbitrary and do not need adjustment. Instead, they are calibrated automatically. The motion capturing workwear is used in a real-world setting, estimating the posture of a worker welding steel on a shipyard. That would not be possible using a motion capturing suit relying on magnetometers.

Contents

Chapter 1

Introduction

This dissertation is about motion capturing with inertial sensors. To capture a motion means to record the orientations, positions, or both of a rigid body or a system of rigid bodies.

The conjunction of the orientation and the position of a rigid body is called the body's *pose*. A *skeleton* is a system of jointed rigid bodies; human bones are the rigid bodies of a human skeleton. The collection of poses of all bodies of a skeleton is the skeleton's *posture*.

In this dissertation, the assumptions necessary to estimate orientations and poses of single rigid bodies from inertial sensor data only are explored. Building upon that, an estimator to obtain the posture of a generic skeleton, whose bodies are jointed by hinges or spherical joints, is devised. The posture estimator is additionally implemented in workwear, to capture the motion of manual workers from inertial sensors only and in real-time.

Motion capturing, that is orientation, pose or posture estimation, is relevant for countless applications in engineering, science and art.

Spacecrafts, rather big rigid bodies, need to estimate their orientation to stabilize and rotate themselves relative to astronomical objects. Submarines use Inertial Navigation Systems to guide themselves. A walking robot uses its trunk's orientation to tell whether or not it is standing upright and to determine in which direction it is walking. Characters in movies are animated very naturally by estimating the posture of an actor. For motion and gait analysis, physiotherapists measure angles of human joints, and thus the relative orientation of two jointed bodies.

Every smartphone owner carries an orientation and position estimator. Smartphones need to know their position to provide location-dependent services, their orientation to correctly display user interfaces and both the orientation and position to be able to provide navigation aids.

There is a big variety of sensors to estimate orientations and positions. To determine positions, the Global Positioning System (GPS) comes to mind almost immediately. However, positions often do not need to be determined globally. If a position to a local reference is good enough, cameras may be used to observe landmarks to determine the position relative to those landmarks. A football player, for instance, needs to know his position on the field, but he does not need to know where exactly the field is on earth. Cameras might also be used to observe the objects themselves, for example markers on the limbs of actors.

Inertial Sensors

Inertial sensors, that is rigidly coupled combinations of gyro- and accelerometers, are hugely popular to estimate orientations. A gyrometer, also called rate gyroscope or simply gyroscope, measures the angular velocity with which the sensor turns. Given an initial orientation, the gyrometer signal can be integrated over time to provide orientation estimates over time.

Integrating measurements means integrating the true value plus some measurement error. Thus, the longer the integration period, the worse the resulting orientation estimate.

An accelerometer measures acceleration. However, if at rest, e. g. lying on the ground, an accelerometer does not measure zero acceleration. Instead, it measures the acceleration the ground exerts on the sensor to keep it from falling through the ground, which is, of course, minus the gravitational acceleration, $-g$. All accelerations measured with an accelerometer are set off by $-g$. An instructive example of this is an accelerometer in free fall, which measures zero, although it clearly accelerates with g. Since $-g$ points upwards, it can be used as an absolute reference for the orientation: Given the current orientation estimate, the negative gravity vector, as it is expected to be measured by the accelerometer, is computed. The difference between this expected negative gravity vector and the actual accelerometer measurement is then used to correct the orientation estimate.

The error derived from gravity can not include the orientation error around the vertical axis, which is parallel to gravity. The rotation around the vertical axis, the *heading*, does not influence the expected gravity vector in local sensor coordinates. To resolve this missing degree of freedom (DOF), a second vector which is both known "globally" (like g) and measurable is necessary. The most well-known example is the earth's magnetic field. This, however, only works if the magnetic field and gravity are not collinear, i. e. it does not work at the magnetic north or south pole of the earth, where the magnetic field is perpendicular to the earth's surface. Practically, it also does not work inside buildings or other structures containing lots of metal, wires or other magnetic disturbances.

Inertial sensors are also useful to estimate complete poses. If the initial pose of an inertial sensor is known, the integrated orientation can be used to add the gravity offset back to the accelerometer measurement, which in turn can be doubly integrated to obtain a position estimate. This is the idea behind an Inertial Navigation System (INS). However, since all inertial measurements are integrated as relative measurements, there is nothing that could correct the also accumulated errors. Thus, a common combination of sensors is INS and GPS.

Restriction on Inertial Sensors and SIRKA

A common technique in orientation or pose estimation is to combine inertial sensors with other sensors, which provide long-term, absolute information about position or orientation.

However, it is interesting to know how far one can get *using inertial sensors only* if there is some prior information. Such prior information may include the knowledge that the moving rigid body

does not move infinitely far away from where it started, e. g. moves around in a room. Or that the bodies, whose poses are to be estimated, are all connected via joints to form a skeleton.

Aside from the theoretical interest, there is also practical value in that restriction. The bigger part of this work has been developed in context of the research project SIRKA, which aimed to develop a "sensor suit for individual activity feedback" (in German: "Sensoranzug zur Rückkopplung körperlicher Aktivität"). The suit estimates postures of manual workers while they are working in previously unknown or very restrictive environments that can not be modified to install external sensors for motion capturing, e. g. in rescue services or shipbuilding, such as in Fig. 1.1. The idea is to enable a worker to present a recording of the postures he assumed during his work to a physiotherapist, who then identifies potentially dangerous postures in the recording. That done, if postures deemed dangerous occur in the real-time estimates, the suit is supposed to warn, e. g. through vibration. Additionally, posture recordings are useful to evaluate the ergonomics of workplaces, i. e. to identify tasks which include particularly many potentially dangerous postures.

At the time of writing, SIRKA is an ongoing project. Aside from the *DFKI* responsible for the sensor fusion, the project partners are *rofa Bekleidungswerk* making the cloths, *Budelmann Elektronik* making the electronics as well as *INAPO*, *Offis*, the *Meyer Werft*, and the *Johanniter Unfallhilfe*. INAPO is a division of the Osnabrück University of Applied Sciences which develops models to judge the harmfulness of postures. These are used at Offis to develop a classifier that uses the postures estimated by the sensor fusion. The Johanniter Unfallhilfe is a rescue service, the Meyer Werft a major shipyard based in Papenburg in northern Germany. Both defined the considered application environment and helped with the technical evaluation, i. e. provided the testing environments, and are responsible for the evaluation concerning the application, i. e. working in the suit to find potentially harmful postures.

It is impossible to cover the entire workspace of, for instance, a shipbuilder with external sensors, such as cameras. Additionally, the magnetic field around a worker continuously changes as he moves in an unfinished ship, due to the ship being mostly made of steel and the tasks which are carried out, such as electric welding.

At the beginning of the project SIRKA, an early prototype of the inertial sensor network was built into workwear of the Meyer Werft. The early prototype still had magnetometers. To test whether the electronic components and the sensor network itself survive in the targeted environment, workers performed example tasks in a steel frame, displayed in Fig. 1.1, that resembles the steel modules used to assemble cruise ships. Figure 1.2 plots the magnetic field experienced by one of the magnetometers. Even before the worker enters his actual workspace, the magnetic field varies considerably. After entering the steel cube, the magnetic field measured has multiple times the strength of the Earth's magnetic field. Even worse, it varies very quickly over multiple Earth magnetic field strengths. Thus, the magnetic field on a shipyard is not usable for orientation estimation.

Aside from mechanical sensors, such as goniometers, which require accurate alignment and are thus hard to set up, inertial sensors are pretty much what is left, with cameras and magnetometers ruled out. Thus, a suit, which estimates the poses of all limbs of the wearer of the suit using inertial measurements only, is desirable. Consequently, the SIRKA suit sparked a lot of interest, prototypes have been featured both on CeBIT 2015 and 2016, and on the Hannover Messe 2016,

Figure 1.1 *Shipyard Environment*
Worker of the Meyer Werft welding in the environment targeted by the project SIRKA. The steel cube itself and the equipment disturb the magnetic field, rendering motion capturing techniques that rely on magnetometer measurements useless. Despite the adverse environment, the SIRKA suit provides an estimate of the worker's posture rendered in the left part of the image. Each pyramid of the rendering represents a limb in the pyramid's direction, whose origin is in the pyramid's base.

the latter on invitation of the German Ministry of Education and Research (BMBF). To my knowledge, the suit appeared in the press 41 times. It also appeared on regional TV in the course of CeBIT coverage. The German science TV show "Wissen vor 8", which runs 5-minute episodes right before prime-time news on national TV, also devoted one of their episodes to a prototype of the suit [ARD].

1.1 Problem Statements

More formally the problems this thesis deals with are the following.

Orientation Estimation

Let there be a fixed coordinate system with three perpendicular coordinate axes of unit length, x_W, y_W and z_W. For example, imagine a world-fixed coordinate system with yourself at the origin, the z_W-axis pointing upwards, i. e. against the direction of gravity, and the x_W-axis pointing northwards. Conventionally, coordinate systems are always right-handed, so the remaining axis is $y_W = z_W \times x_W$, pointing westwards.

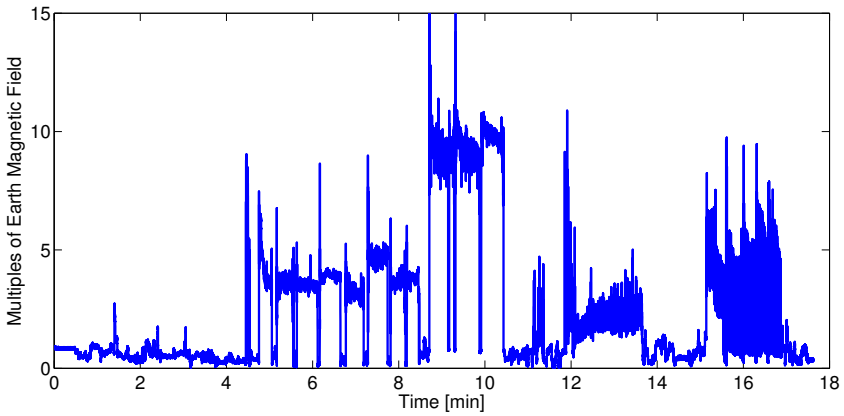


Figure 1.2 *Magnetic Field on a Shipyard*
The plot shows norms of measurements of a magnetometer which was carried by a worker of the Meyer Werft while performing example tasks. After a little more than four minutes, the worker enters a steel cube similar to a module of a cruise ship. While the worker performs various tasks, including electric welding, the magnetic field changes frequently, becoming more than 15 times stronger than the Earth's magnetic field.

The local coordinate system is defined similarly, with axes x_B, y_B, and z_B fixed on the body whose orientation is to be estimated. x_B points in the body's forward direction, z_B points in the body's upward direction and $y_B = z_B \times x_B$ points to the left, completing the right-handed coordinate system.

The orientation of this body relative to the world-fixed coordinate system is described by a mapping which represents the body axes using the world-fixed axes. The mapping must maintain the unit length of the axes, the right-handedness and the pairwise perpendicularity.

Such a mapping is easily represented by a rotation matrix, whose columns are the axes of the body expressed in world coordinates:

$$Q_{W\urcorner B} = \begin{bmatrix} x_{W\urcorner B} & y_{W\urcorner B} & z_{W\urcorner B} \end{bmatrix} \tag{1.1}$$

An example is given in Fig. 1.3. Formally, $Q_{W\urcorner B}$ is an element of the three-dimensional special orthogonal group, i. e. $Q_{W\urcorner B} \in \{Q \in \mathbb{R}^{3\times 3} | Q^T Q = QQ^T = I, \det Q = 1\} = \mathbb{SO}(3)$. I is the identity matrix.

$Q_{W\urcorner B}$ can be used to transform any relative vectorial quantity represented in body coordinates to its equivalent representation in world coordinates. Here, *relative* means that the quantity is independent of the coordinate systems' origins. Examples of such relative quantities are linear velocities and accelerations.

For example, let the gravitational acceleration expressed in body coordinates be g_B. Using eq. (1.1), we can calculate the gravitational acceleration in world coordinates, $g_W = Q_{W\urcorner B} g_B$.

The orientation estimation problem is to determine $Q_{W\urcorner B}$ from a set of n vectors which are known or measured both in W-coordinates and in B-coordinates, $\{u_W^{(0)}, u_B^{(0)}, \ldots, u_W^{(n-1)}, u_B^{(n-1)}\}$,

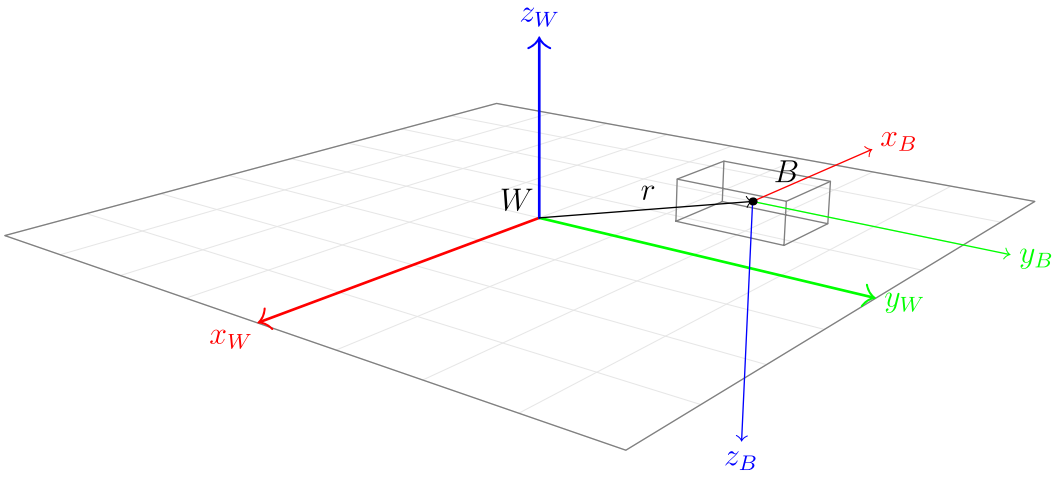


Figure 1.3 *World and Body coordinate systems.*
The orientation of the body B (grey cube) is relative to the fixed reference or world coordinate system W (plane). The orientation of B in world coordinates is represented by a rotation matrix $Q_{W\urcorner B} \in \mathbb{SO}(3)$, whose columns are the axes of the B-coordinates expressed in terms of the axes of the world coordinates. In this case, the local coordinate system of B is rotated 180° around y_W, so the rotation matrix is $Q_{W\urcorner B} = \begin{bmatrix} -x_W & y_W & -z_W \end{bmatrix}$. The pose of B also includes the position r of its origin, also expressed in world coordinates.

such that

$$\text{for } 0 \leq k < n : u_W^{(k)} = Q_{W\urcorner B} u_B^{(k)}. \tag{1.2}$$

This can be extended to the dynamic orientation estimation problem by allowing the coordinate systems to change. Eq. (1.2) becomes

$$\text{for } 0 \leq k < n : u_W^{(k)} = Q_{W\urcorner B}^{(k)} u_B^{(k)} \quad \text{s.t. for } 1 \leq k < n : {Q_{W\urcorner B}^{(k-1)}}^{-1} Q_{W\urcorner B}^{(k)} = \Delta Q^{(k)}, \tag{1.3}$$

where the orientation changes $\Delta Q^{(k)}$ are either known or measured.

Estimating the orientation of a moving, i. e. rotating, body using inertial sensors and magnetometers with respect to a world-fixed reference frame is a dynamic orientation estimation problem: Quantities known in world coordinates such as the earth's magnetic field and gravity are measured in body coordinates using magnetometers and accelerometers. The orientation changes are measured using gyroscopes.

This work is mostly concerned with the dynamic orientation estimation problem. Also, the interest is not in orientations only. Instead, the orientation is to be estimated in conjunction with sensor calibration quantities, e. g. the bias of the gyroscope, or additional quantities about the trajectory, as stated below.

Pose Estimation

Orientation estimation can be thought of as a subproblem of pose estimation. In addition to the orientation, the pose also includes the displacement between the origins of the coordinate systems.

With the orientation $Q_{W\urcorner B}$ and the origin $r_{W\urcorner B}$ of the B-coordinate system expressed in the W-coordinate system, any point v_B in B-coordinates (not only relative quantities that do not depend on the origin), can be transformed to W-coordinates:

$$v_W = r_{W\urcorner B} + Q_{W\urcorner B} v_B \tag{1.4}$$

This is often conveniently written as

$$\begin{bmatrix} v_W \\ 1 \end{bmatrix} = \begin{bmatrix} Q_{W\urcorner B} & r_{W\urcorner B} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_B \\ 1 \end{bmatrix}, \tag{1.5}$$

where $T_{W\urcorner B} = \begin{bmatrix} Q_{W\urcorner B} & r_{W\urcorner B} \\ 0 & 1 \end{bmatrix}$ is called the homogeneous coordinate transform. Relative vectors, such as velocities, that do not depend on a point, can also be transformed using $T_{W\urcorner B}$. Such vectors are extended with a 0 instead of a 1. When multiplying out an expression such as eq. (1.6), it reduces to the multiplication of the vector with the rotation matrix.

$$\begin{bmatrix} v_W \\ 0 \end{bmatrix} = \begin{bmatrix} Q_{W\urcorner B} & r_{W\urcorner B} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_B \\ 0 \end{bmatrix} = \begin{bmatrix} Q_{W\urcorner B} v_B \\ 0 \end{bmatrix} \tag{1.6}$$

The pose estimation problem is to find both $Q_{W\urcorner B}$ and $r_{W\urcorner B}$ or equivalently $T_{W\urcorner B}$.

Posture Estimation

For the collection of bodies, e. g. bodies A, B, and C, which are connected by joints, e. g. a joint between A and B and another one between B and C, the posture estimation problem is to find the poses of all bodies. In this example these are $T_{W\urcorner A}$, $T_{W\urcorner B}$ and $T_{W\urcorner C}$.

This can be restated as finding the global pose of one body only, $T_{W\urcorner A}$, and the relative poses between adjacent bodies, which are connected by joints, $T_{A\urcorner B}$ and $T_{B\urcorner C}$. The poses of all bodies in world coordinates are easily reconstructed by multiplying the matrix representations of the relative poses:

$$T_{W\urcorner B} = T_{W\urcorner A} T_{A\urcorner B} \quad T_{W\urcorner C} = T_{W\urcorner A} T_{A\urcorner B} T_{B\urcorner C} \tag{1.7}$$

This restatement can be used to simplify the posture estimation problem. In this work, it won't matter where the skeleton is positioned globally. So instead of estimating the global position of body A, it is defined to be just zero. Additionally, the position of a relative pose is (almost) constant, e. g. the position of the elbow in shoulder coordinates does not change, because the

bones between shoulder and elbow are (almost) rigid. Consequently, in posture estimation only the relative orientations have to be estimated over time.

> To summarize, it all boils down to is this fundamental problem:
>
> Find $Q^k_{A\urcorner B}$ and, if necessary and not otherwise known, $r^{(k)}_{A\urcorner B}$, from $\Delta Q^{(k)}$ and pairs of vectors $u^{(k)}_A$ and $u^{(k)}_B$ in A- and B-coordinates, where $u^{(k)}_A$, $u^{(k)}_B$ and $\Delta Q^{(k)}$ can either be measured or are known from prior information.

1.2 Contribution

There are two slightly different pose estimation tasks:

1. finding the pose of a rigid body equipped with sensors relative to a fixed reference, usually the world reference frame with the vertical axis pointing from the ground upwards, and one horizontal axis pointing from south to north, and
2. finding the pose of a rigid body equipped with sensors relative to another rigid body, also equipped with sensors. This is called a *relative* pose.

To estimate the posture of a human, solving task 2 for all pairs of bodies which are connected over a joint is almost sufficient. In addition to the relative positions and relative orientations of all bodies, usually the global, task-1-like inclination of each body should be estimated. Without that, it would be impossible to tell whether the skeleton is standing upright or lying on the ground, for instance.

This work contributes to the solution of both tasks.

Task 1: "Global" Poses

The accelerometer measurement is the second derivative of the position plus an offset, negative gravity. It is used accordingly as a relative, dynamic measurement in INS. It is curious, that the same measurement appears as an *absolute* measure for the inclination error in orientation estimation. In contrast to, for instance, the magnetometer measurement, it is clearly not an absolute measurement. This work includes

- a theoretical analysis of the assumption justifying this practice and its formalization as a probabilistic prior,
- two alternative priors, yielding new orientation estimators which are validated experimentally together with a proof of the validity of the way time-correlations are handled,
- an extension of one of the new estimators to provide a complete 6-DOF pose and
- the reason why the other two orientation estimators can not be extended the same way.

The orientation of the so estimated pose still is globally accurate and its position is locally as accurate as an integrated position but does not drift unboundedly. While this might be practically relevant, the main focus lies in understanding of what happens when inertial sensor data and prior information are fused.

Task 2: Relative Poses / Posture

Estimating the task-1-like orientation of a single body with inertial sensors only is impossible. The rotation around the vertical axis can not be determined without additional aids, such as a magnetometer.[1] The posture of a skeleton, with all three degrees of freedom of each relative orientation, including rotation around the vertical axis, however, can be determined with inertial sensors only, *if the skeleton moves*. The main practical contribution includes

- an algorithm to estimate the relative poses of the bodies of a skeleton, e. g. a human's posture, from inertial sensor data only, without other sensors such as magnetometers and
- a technique to implement this algorithm on computationally constrained devices by decoupling the estimation from the sampling rate of the sensors.

The algorithm has been implemented on such a computationally constrained device for the research project SIRKA, which was already mentioned.

The actual suit is probably a contribution at least as big as this book. The suit is not only an example implementation of a posture estimator using inertial sensor data only. It is also a tool to assess the ergonomics of workplaces, that were almost unobservable previously.

1.3 Outline

The state of the art related to this work is reviewed in Chap. 2 with an emphasis on orientation estimation. It is the basic problem which underlies pose estimation for a single rigid body (task 1) and posture estimation (task 2) of skeletons, e. g. systems of jointed rigid bodies. Aside from the question about the assumptions necessary to estimate orientations from inertial sensor data, the question how to estimate an orientation in the first place, independent of the kind of measurements, has to be answered. Various strategies to solve the problem have been developed with different advantages. Other works of posture or joint angle estimators are also covered. Although it plays almost no role in this dissertation, biomechanics is mentioned very briefly.

Chapter 3's topic is pose estimation for a single rigid body using inertial sensor data only, i. e. task 1.

[1] This is not strictly true. Gyroscopes, such as fibre optic gyroscopes, which are accurate enough to measure the Earth's rotation (and thus its rotation axis), do exist. It is also possible with MEMS sensors [Ioz+12] not entirely unlike the sensors used here, but only with a much smaller measurement range, a rather complex hardware setup and very slowly. With sensors embedded in a suit, which are also supposed to measure human motions, it is not a practical possibility.

Building on a result of Chap. 3, Chap. 4 and 5 are mainly concerned with task 2, estimating the posture of a skeleton, again using inertial sensor data only. Chap. 4 covers the estimation algorithms to build a motion capturing system for a generic skeleton, not necessarily human. Chap. 5 covers their implementations for SIRKA, the motion capturing workwear.

Chapter 6 concludes.

Chapter 2

State of the Art

Various methods to determine orientations and poses of rigid bodies exist. Some are based on generic estimation algorithms, others are specialized estimators for their task. The algorithms to choose from are reviewed in Sect. 2.1, the ones which are actually employed in this work are explained in detail.

Different strategies to obtain joint angles or postures involving inertial sensors have already been developed. They are reviewed in Sect. 2.2.

Some of these have been used in existing sensor suits not entirely dissimilar to SIRKA. These are subject of Sect. 2.3. For completeness, some efforts in biomechanical modeling are mentioned in Sect. 2.4, although the models used in this work are much more simplistic.

2.1 (Orientation) Estimation

The two main issues in estimating orientations are non-linearity and the representation of the orientation itself. Although an orientation has only 3 DOF, 3-dimensional representations of orientations are not suitable for generic estimation algorithms, because all suffer from singularities [Stu64]. This leads to situations in which a very small change in the actual orientation leads to a large change in the representation, or a change in the representation leads to almost no change in the actual orientation. An instructive example is a 1-dimensional orientation represented by an angle α. Changing the orientation a little in most cases changes α a little. In a bad case, however, α might wrap from $-\pi$ to π, resulting in a big numerical change.

Kalman Filter

To solve both the non-linearity and the singularity issues, a number of algorithms have been developed [CMC07]. Most of these algorithms are based on the Kalman Filter, a well known estimation algorithm originally discovered by R.E. Kalman [Kal60].

Let there be a linear dynamic discrete-time system with state $x^{(t)}$ at time step t, input $u^{(t)}$, state transition matrix $A^{(t)}$, input gain $B^{(t)}$, and Gaussian process noise $v^{(t)} \sim \mathcal{N}\left(0, \Sigma_z^{(t)}\right)$, as in eq. (2.1)

$$x^{(t)} = A^{(t)}x^{(t-1)} + B^{(t)}u^{(t)} + v^{(t)} \ . \tag{2.1}$$

Further, let the measurements $z^{(t)}$ of the system be linearly related to its state by the measurement matrix $H^{(t)}$ and also subject to Gaussian noise $w^{(t)} \sim \mathcal{N}\left(0, \Sigma_p^{(t)}\right)$, i.e.

$$z^{(t)} = H^{(t)}x^{(t)} + w^{(t)} \ . \tag{2.2}$$

Given an initial state drawn from a normal distribution, $x^{(0)} \sim \mathcal{N}\left(\hat{x}^{(0)}, \Sigma^{(0)}\right)$ with expectation $\hat{x}^{(0)} = E[x^{(0)}]$, all further states $x^{(t)}$ and all measurements $z^{(t)}$ are normally distributed. The discrete time Kalman Filter [BKL02, chapter 5] is a recursive algorithm to compute the expectation $\hat{x}^{(t)} = E[x^{(t)}|z^{(1:t)}, u^{(1:t)}]$ and covariance $\Sigma^{(t)}$ of time step t given all inputs and measurements up to time step t:

$$\hat{x}^{(t)} = \bar{x}^{(t)} + K^{(t)}(z^{(t)} - H^{(t)}\bar{x}^{(t)}) \tag{2.3}$$

$$\Sigma^{(t)} = \bar{\Sigma}^{(t)} - K^{(t)}H^{(t)}\bar{\Sigma}^{(t)}, \tag{2.4}$$

where $K^{(t)} = \bar{\Sigma}[H^{(t)}]^T(H^{(t)}\bar{\Sigma}[H^{(t)}]^T + \Sigma_z^{(t)})^{-1}$ is called the Kalman Gain. The intermediate expectation $\bar{x}^{(t)} = E[x^{(t)}|z^{(1:t-1)}, u^{(1:t)}]$ and the corresponding covariance $\bar{\Sigma}^{(t)}$ are computed using the system dynamics from the expectation and covariance of the previous time step:

$$\bar{x}^{(t)} = A^{(t)}\hat{x}^{(t-1)} + B^{(t)}u^{(t)} \tag{2.5}$$

$$\bar{\Sigma}^{(t)} = A^{(t)}\Sigma^{(t-1)}{A^{(t)}}^T + B^{(t)}\Sigma_u^{(t)}{B^{(t)}}^T + \Sigma_p^{(t)} \tag{2.6}$$

Eq. (2.6) accounts for the possibility that the input may also be a random variable subject to Gaussian noise, i. e. $u^{(t)} = s^{(t)} + r^{(t)}$, $r^{(t)} \sim \mathcal{N}\left(0, \Sigma_u^{(t)}\right)$, for some true input $s^{(t)}$. If that is not a case and $u^{(t)}$ is not a random variable, $B^{(t)}\Sigma_u^{(t)}{B^{(t)}}^T$ in eq. (2.6) vanishes.

Extended Kalman Filter

For non-linear measurement models, $z^{(t)} = h(x^{(t)}) + w^{(t)}$, and non-linear dynamic models, $x^{(t)} = f(x^{(t-1)}, u^{(t)}) + v^{(t)}$, the Kalman Filter can still be used. Eq. (2.5) and (2.3) become

$$\bar{x}^{(t)} = f(\hat{x}^{(t-1)}, u^{(t)}) \quad \text{and} \quad \hat{x}^{(t)} = \bar{x}^{(t)} + K^{(t)}(z^{(t)} - h(\bar{x}^{(t)})) \ . \tag{2.7}$$

The rest remains unchanged, with the matrices being the Jacobians of the model functions evaluated at the current estimate, i.e.

$$A^{(t)} = \frac{\partial}{\partial x} f(x, u^{(t)})\bigg|_{x=\hat{x}^{(t-1)}} \quad B^{(t)} = \frac{\partial}{\partial u} f(x^{(t-1)}, u)\bigg|_{u=u^{(t)}} \quad H^{(t)} = \frac{\partial}{\partial x} h(x)\bigg|_{x=\bar{x}^{(t)}} . \tag{2.8}$$

For linear f and h, this is the same as the original Kalman Filter. If one of the model functions is non-linear, the new algorithm is called the Extended Kalman Filter (EKF) [BKL02, chapter 10].

If the models are linear and all noise is Gaussian, the Kalman Filter is an optimal minimum mean square error estimator [BKL02, chapter 5], i. e. it is guaranteed to find

$$\hat{x}^{(t)} = \arg\min_{\hat{x}^{(t)}} E\left[(\hat{x}^{(t)} - x^{(t)})^2 | u^{(1:t)}, z^{(1:t)}\right] \quad . \tag{2.9}$$

The EKF does not give such guarantees. However, if the error is very small, a lot of non-linear models become locally linear enough such that the EKF yields satisfactory results.

Kalman Filter based Orientation Estimators

Consequently, the EKF has been used to estimate orientations. For example in [Ida96], Rodrigues or scaled-axis parameters of an orientation matrix are estimated from the vector measurement model

$$v = Q(q + \delta q)u \tag{2.10}$$

where q is the three orientation parameters to be estimated, δq the orientation error and $v, u \in \mathbb{R}^3$ are the corresponding vectors in local sensor coordinates and in a global reference frame. Using partial derivatives of Q, δq is stated in terms of the difference of the vectors and the current estimate, which is then massaged into the EKF measurement update.

This algorithm appears to work quite well as long as q does not get close to a singularity. More importantly, for the three degrees of freedom, there is only a 3×3 covariance matrix.

An algorithm which fails whenever the orientation happens to get close to a singularity is not applicable to estimate arbitrary poses of bodies with arbitrary orientations. One way to fix this is to use a redundant orientation representation: Instead of the orientation estimate itself, three parameters, $\delta\hat{q}$, estimated by the EKF describe an error rotation matrix, $Q(\delta\hat{q})$, which is the small difference between the predicted orientation, $\bar{Q}$, and the true orientation: $Q = \bar{Q}Q(\delta\hat{q})$. Because the algorithm has to maintain both $\bar{Q}$ and $\delta\hat{q}$, which are multiplied to yield a useful orientation estimate, the algorithm is called Multiplicative EKF (MEKF) [CMC07]. Using quaternions instead of rotation matrices the algorithm is derived in detail by Lefferts et al. [LMS82]. Notably, they compare the MEKF to an EKF that estimates the orientation quaternion directly. The latter causes problems because the estimator has to maintain a structure of the state which it does not know about per se, i. e. it has to maintain 4 numbers which must make up a unit quaternion and have a covariance of rank 3 instead of 4.

In the MEKF, similar to the 3-parameter EKF with the singularity issue, the covariance of $\delta\hat{q}$ has conceptual meaning, i. e. the orientation uncertainty, though in case of the MEKF in sensor-local coordinates instead of world coordinates.

Unscented Kalman Filter

Instead of computing the derivatives of the dynamic and measurement models at the current estimate, the derivatives can be approximated by sampling the models at representative points, called sigma points, for the (Gaussian) distribution of the current estimate, e. g. at the current estimate and one standard deviation away from the current estimate along and against each dimension. This is the idea behind the Unscented Kalman Filter (UKF) [JU97].

The algorithm to propagate the Gaussian distribution $\mathcal{N}(\mu_x, \Sigma_x)$, $\mu_x \in \mathbb{R}^N$, through a non-linear function f and to approximate the result as a Gaussian distribution $\mathcal{N}(\mu_y, \Sigma_y)$, explained in detail in [JU97], is as follows.

Let $\sqrt{\Sigma_x}_i$ be the i'th dimension of the matrix square root, such as the i'th column of the lower-triangular matrix, $\sqrt{\Sigma_x}_i = L_{(:,i)}$, of the Cholesky decomposition $\Sigma_x = LL^T$ (although any matrix square root decomposition would work). The corresponding set of sigma points is

$$\mathrm{sp}(\mu_x, \Sigma_x) = \mathcal{X} = \{\mu_x\} \cup \left\{\mu_x + \sqrt{\Sigma_x}_i,\ \ \mu_x - \sqrt{\Sigma_x}_i\right\}_{i=1}^{N} \tag{2.11}$$

The corresponding propagated sigma points are simply the elements of the image of $\mathcal{X}$ through the (non-linear) function f,

$$f(\mathcal{X}) = \mathcal{Y} = \left\{y^{(i)} = f(x^{(i)}) | x^{(i)} \in \mathcal{X}\right\} \text{ (with } 1 \leq i \leq 2N+1\text{).} \tag{2.12}$$

From $\mathcal{Y}$, the expected value and covariances are computed by

$$\mu(\mathcal{Y}) = \frac{1}{2N+1} \sum_{i=1}^{2N+1} y^{(i)} \quad \Sigma(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \sum_{i=1}^{2N+1} \left(x^{(i)} - \mu(\mathcal{X})\right) \left(y^{(i)} - \mu(\mathcal{Y})\right)^T \ . \tag{2.13}$$

$\mu_y = \mu(\mathcal{Y})$ and $\Sigma_y = \Sigma(\mathcal{Y}, \mathcal{Y})$ are the expected value and covariance of the normal distribution approximating the original normal distribution $\mathcal{N}(\mu_x, \Sigma_x)$ propagated through the non-linear function, which is not necessarily Gaussian. The mean as computed in eq. (2.13) is accurate up to second order, whereas the EKF yields an only first-order accurate mean [JU97]. Also note that the sigma points do not have to be exactly one standard deviation off the mean if the sigma points in eq. (2.13) are weighted differently, again as described in [JU97].

The Unscented Kalman Filter uses the sigma point propagation through the system dynamics function to compute the intermediate distribution parameters, $\bar{x}^{(t)}$ and $\bar{\Sigma}^{(t)}$. Similarly, the sigma points are propagated through the measurement model to compute the expected measurement, its covariance and its cross-covariance with the intermediate quantity. If a function depends on more than one Gaussian random variable, the variables are simply stacked into a vector. (Conceptually this is nothing new, because the sigma point propagation assumes a multivariate Gaussian anyway.)

Thus, the intermediate expectation and covariance, i. e. the Kalman Filter equations (2.5)

and (2.6), become computing and propagating the sigma points,

$$\mathcal{X}^{(t)} = \mathrm{sp}\left(\begin{bmatrix}\hat{x}^{(t-1)} \\ u^{(t)}\end{bmatrix}, \begin{bmatrix}\Sigma^{(t-1)} & 0 \\ 0 & \Sigma_u^{(t)}\end{bmatrix}\right), \quad \bar{\mathcal{X}}^{(t)} = f\left(\mathcal{X}^{(t)}\right), \tag{2.14}$$

and taking their mean and covariance from eq. (2.13),

$$\bar{x}^{(t)} = \mu\left(\bar{\mathcal{X}}^{(t)}\right), \quad \bar{\Sigma}^{(t)} = \Sigma\left(\bar{\mathcal{X}}^{(t)}, \bar{\mathcal{X}}^{(t)}\right) \quad . \tag{2.15}$$

The updated expectation and covariance are obtained by propagating the intermediate sigma points through the measurement model h, and computing the corresponding mean and covariances,

$$\mathcal{Z} = h(\bar{\mathcal{X}}^{(t)}) \tag{2.16}$$

$$\hat{x}^{(t)} = \bar{x}^{(t)} + K^{(t)}\left(z^{(t)} - \mu\left(\mathcal{Z}\right)\right) \tag{2.17}$$

$$\Sigma^{(t)} = \bar{\Sigma}^{(t)} - K^{(t)}\Sigma\left(\mathcal{Z}, \bar{\mathcal{X}}^{(t)}\right) \tag{2.18}$$

$$\text{with } K^{(t)} = \Sigma\left(\mathcal{Z}, \bar{\mathcal{X}}^{(t)}\right)^T \left[\Sigma\left(\mathcal{Z}, \mathcal{Z}\right) + \Sigma_z^{(t)}\right]^{-1} \quad . \tag{2.19}$$

In principle, any of the strategies to estimate orientations using an EKF also works with an UKF, for example maintaining the 3-dimensional orientation error separately [CMC07].

However, Kraft [Kra03] extended the UKF to estimate an orientation quaternion directly while retaining its structure. Kraft puts the orientation unit quaternion into the state vector to be estimated. This introduces lots of problems, e. g. with the sigma point computation in eq. (2.11) or the calculation of mean and covariance in eq. (2.13). In general, including a manifold element in the state vector breaks the algorithm steps using vectorial addition (or subtraction) involving these elements because, for instance, the difference of two unit quaternions is not necessarily a unit quaternion. Kraft works around these issues by using quaternion multiplication and inversion whenever a change in orientation has to be obtained or "added" to the quaternion component of the state. Since these changes are small, they are also safely, i. e. far away from a singularity, represented as a 3-dimensional vector, on which vectorial sums can be computed safely.

This is a major achievement, because it lets an essentially generic estimation algorithm for vectorial quantities estimate a manifold element without singularity issues or non-linear constraints. Kraft's modifications to the UKF can be generalized to let generic algorithms estimate elements of certain manifolds, as is explained a further down in this chapter.

Specialized Orientation Estimators

Although Kalman Filters appear to be almost ubiquitous, there are algorithms solving the orientation estimation problem of eq. (1.2), that are not based on the Kalman Filter.

Wahba [Wah65] restated eq. (1.2) as a cost function

$$\arg\min_{Q_{W\urcorner B}} L(Q_{W\urcorner B}) \text{ with } L(Q_{W\urcorner B}) = \sum_{k=0}^{n-1} \|u_W^{(k)} - Q_{W\urcorner B} u_B^{(k)}\|^2 \ . \tag{2.20}$$

This minimum can, for example, be found using the singular value decomposition [Mar88]. According to Markley, the function to minimize in eq. (2.20) can be rewritten as

$$L(Q_{W\urcorner B}) = 1 - \operatorname{tr}\left(Q_{W\urcorner B} B^T\right) \text{ with } B = \sum_{k=0}^{n-1} u_W^{(k)} [u_B^{(k)}]^T \tag{2.21}$$

with the following solution to the minimization problem:

$$Q_{W\urcorner B} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det U \det V \end{bmatrix} V, \tag{2.22}$$

where U and V are the orthonormal matrices of the singular value decomposition of $B = USV$.

There are more solutions to this problem. The survey by Crassidis et. al [CMC07] mentions the q-Method in particular, where maximizing $\operatorname{tr}\left(Q_{W\urcorner B} B^T\right)$ is recast to finding the eigenvector, q, which corresponds to the quaternion representation of $Q_{W\urcorner B}$, with the largest eigenvalue of the matrix K which satisfies $\operatorname{tr}\left(Q_{W\urcorner B} B^T\right) = q^T K q$. A modification of the q-Method, which finds the largest eigenvalue of K iteratively, is called *QUEST*, for Quaternion Estimator.

To adapt these methods to the dynamic estimation problem, fading memory approaches have been used, which are, however, considered inferior to Kalman-Filter-based algorithms [CMC07].

An algorithm called Extended QUEST by Psiaki [Psi00; CMC07] solves this problem and also allows to include additional variables in the estimation problem. It introduces dynamic models for the quaternion and additional variables and information matrices, which are used in an extension of the cost function in eq. (2.21), $L(Q_{W\urcorner B}, x)$, where x is the vector of additional variables. The information matrices are used to introduce a weighted penalty in the cost function for changing x or $Q_{W\urcorner B}$ away from the current estimate.

The algorithm itself is a (square-root) information filter, which propagates the estimates of $Q_{W\urcorner B}$ and x using the dynamic models and propagates the information matrices using linearizations of the dynamic models. In the measurement update, $L(Q_{W\urcorner B}, x)$ is solved using the propagated estimates and matrices for the optimal $Q_{W\urcorner B}$. Using the optimal $Q_{W\urcorner B}$, $L(Q_{W\urcorner B}, x)$ is solved for the new estimate of the auxiliary variables x. The details given in [Psi00] are a little involved. The key point, however, is that the computation of the optimal $Q_{W\urcorner B}$ is still a direct solution of Wahba's problem.

Thus, in contrast to generic estimation algorithms like the EKF, Extended QUEST has built-in knowledge about the structure of the orientation, which leads to fabulous convergence with respect to the orientation: the estimator converges from any initial orientation error [Psi00] (which is at worst 180° due to periodicity). It also avoids the singularity problem the 3-parameter

EKF has, and does not require maintaining two orientation quantities, which is what the MEKF does.

Another approach to solve the filtering problem is to use the Bingham distribution [Bin74], which is essentially the distribution of a multivariate Gaussian random variable conditioned to be on a unit sphere, i. e. it appears to be the natural distribution of a random unit quaternion. Filters structurally resembling the EKF [GK13] as well as the UKF [Gil+16] have been developed. The maximum likelihood estimate of the Bingham distribution's parameters for a set of samples has exactly the structure of QUEST to solve Wahba's problem. However, it is unclear how to apply the Bingham filters to estimate states consisting of multiple orientations or both Bingham and Gaussian distributed components.

Compared to approaches like Kraft's, Bingham filters are particularly good when the angular uncertainty is large. However, when estimating orientations from inertial sensor data, uncertainty is usually small, once an accurate initial estimate is found.

Manifold Element Estimation

An alternative strategy to building knowledge about the structure of the orientation representation into the estimator is hiding the structure from it entirely by encapsulation. The only places, where generic algorithms like the Extended or Unscented Kalman Filter need to know the state vector itself, are the dynamic and measurement model functions. But, like the contents of the state vector, they are specific to the estimation problem and not generic. The algorithm itself only needs to compute and apply small changes to the state vector. The problem is that doing this with vectorial $+$ and $-$ operators destroys the the structure of orientations (or other non-vectorial manifold elements) in the state.

⊞-Manifolds Hertzberg et al. [Her+13] solved these problems for a subclass of manifolds, called ⊞-manifolds. They introduced two new operators, ⊞ and ⊟, which are used to "add" vectorially represented changes to manifold elements (⊞) and to obtain vectorially represented differences between two manifold elements (⊟).

Let $\mathbb{M}$ be a manifold with n degrees of freedom, for instance $\mathbb{M} = \mathbb{SO}(3)$ with $n = 3$. The two corresponding operators are

$$\boxplus : \mathbb{M} \times \mathbb{R}^n \to \mathbb{M} \quad \boxminus : \mathbb{M} \times \mathbb{M} \to \mathbb{R}^n \ . \tag{2.23}$$

The operators satisfy the following properties for $x \in \mathbb{M}, V \subset \mathbb{R}^n$:

$$x \boxplus 0 = x \qquad \forall y \in \mathbb{M} : x \boxplus (y \boxminus x) = y \qquad \forall \delta \in V : (x \boxplus \delta) \boxminus x = \delta \tag{2.24}$$

That is, ⊞ and ⊟ are inverses of each other. $V \in \mathbb{R}^n$ is the set of unique parameterizations of the manifold around 0. E. g. if $\mathbb{M}$ were 1-dimensional angles, then $V = \{\alpha | -\pi \leq \alpha < \pi\} \subset \mathbb{R}$. Additionally, the distance between two manifold elements must not be larger than the distance

of the vectorial parameterizations to produce them:

$$\forall \delta_1, \delta_2 \in \mathbb{R}^n : \|(x \boxplus \delta_1) \boxminus (x \boxplus \delta_2)\| \leq \|\delta_1 - \delta_2\| \tag{2.25}$$

Hertzberg et al. also showed that Cartesian products, i. e. compounds, of $\boxplus$-manifolds are again $\boxplus$-manifolds. If elements of different $\boxplus$-manifolds $x_1 \in \mathbb{M}_1$, $x_2 \in \mathbb{M}_2$ with operators $\boxplus_1$, $\boxminus_1$ and $\boxplus_2$, $\boxminus_2$, respectively, are combined into a so-called compound manifold $\mathbb{M} = \mathbb{M}_1 \times \mathbb{M}_2$, they induce new operators which use the existing operators on the respective components. With n_1 and n_2 being the degrees of freedom of $\mathbb{M}_1$ and $\mathbb{M}_2$, $\delta_1 \in \mathbb{R}^{n_1}$, $\delta_2 \in \mathbb{R}^{n_2}$

$$\boxplus : \mathbb{M} \times \mathbb{R}^{n_1+n_2} \to \mathbb{M} \quad \text{with} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \boxplus \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} x_1 \boxplus_1 \delta_1 \\ x_2 \boxplus_2 \delta_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \tag{2.26}$$

$$\boxminus : \mathbb{M} \times \mathbb{M} \to \mathbb{R}^{n_1+n_2} \quad \text{with} \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \boxminus \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \boxminus_1 x_1 \\ y_2 \boxminus_2 x_2 \end{bmatrix} = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} \quad . \tag{2.27}$$

Thus, $\boxplus$-manifold elements can be compounded to build a specific estimator state. Note that a vector space with ordinary $+$ and $-$ is also a $\boxplus$-manifold. Although (compound) $\boxplus$-manifolds are not necessarily vector spaces, I write their elements in vector notation. This keeps familiar algorithms, which usually work on vectors, look familiar, even after they have been lifted to $\boxplus$-manifolds. Moreover, $\boxminus$-differences of such elements actually are vectors, which turn out to be the important quantities in most algorithms.

Gaussians on $\boxplus$-Manifolds To define the Gaussian distribution for an n-dimensional $\boxplus$-manifold, Hertzberg et al. use the manifold mean $\mu \in \mathbb{M}$ as a reference point and add the vectorially represented, zero-mean uncertainty according to the covariance Σ using $\boxplus$, that is

$$\mathcal{N}(\mu, \Sigma) \;\hat{=}\; \mu \boxplus \mathcal{N}(0, \Sigma) \quad . \tag{2.28}$$

If the manifold is a vector space, $\mathbb{M} = \mathbb{R}^n$, eq. (2.28) holds exactly. If it is not, then eq. (2.28) is an approximation, which worsens the larger the uncertainty becomes.

To be useful for an Unscented Kalman Filter, the mean and covariance of points drawn from such a distribution have to be computable, as in eq. (2.13). For the covariance, the only change is to replace $-$ by $\boxminus$, since the $\boxminus$-difference is vectorial. The summation to calculate the mean, however, can not be carried out using $\boxplus$, because $\boxplus$ does not add two manifold elements.

Hence, the expected value of a $\boxplus$-manifold random variable is defined as the element with the minimum squared distance to the true value $X \in \mathbb{M}$

$$\mathrm{E}[X] = \arg\min_{x \in \mathbb{M}} \mathrm{E}\left[\|X \boxminus x\|^2\right] \quad . \tag{2.29}$$

Hertzberg et al. compute the expectation using the following recursion.

$$\mathrm{E}[X] = \lim_{k \to \infty} \mu_k \quad \text{with } \mu_{k+1} = \mu_k \boxplus \mathrm{E}[X \boxminus \mu_k] \tag{2.30}$$

Thanks to eq. (2.25), the recursion is guaranteed to converge. In practical implementations, the computation of the limit terminates when $\| \operatorname{E}[X \boxminus \mu_k] \|_2$ falls below a threshold. The definition of the covariance looks very familiar. For two manifold-element random variables $X \in \mathbb{M}_1$ and $Y \in \mathbb{M}_2$

$$\operatorname{Cov}[X, Y] = \operatorname{E}\left[(X \boxminus \operatorname{E}[X]) (Y \boxminus \operatorname{E}[Y])^T \right] \quad . \tag{2.31}$$

Note that X and Y may be elements of different manifolds $\mathbb{M}_1$, $\mathbb{M}_2$, and that both the covariance and expectation on $\boxplus$-manifolds are equivalent to the definitions for multivariate Gaussians, if $\mathbb{M}_1$ and $\mathbb{M}_2$ are vector spaces.

Sigma Point Propagation on $\boxplus$-manifolds Sigma points of a $\boxplus$-manifold random variable are generated by replacing the vector addition in eq. (2.11) with $\boxplus$:

$$\operatorname{sp}_{\boxplus}(\mu_x, \Sigma_x) = \mathcal{X} = \{\mu_x\} \cup \left\{ \mu_x \boxplus \sqrt{\Sigma_x}_i, \;\; \mu_x \boxplus -\sqrt{\Sigma_x}_i \right\}_{i=1}^{N} \tag{2.32}$$

The expected value and covariance of sigma points are calculated according to equations (2.30) and (2.31). The following algorithm using these equations is given by Hertzberg et al. in [Her+13] in a slightly different form. Given a convergence threshold $\epsilon \in \mathbb{R}$ and a set of manifold-element sigma points, $\mathcal{Y} = \{y_0, \ldots, y_{2n}\}$, where y_0 is the (propagated) old mean,

$$\mu_{\boxplus}(\mathcal{Y}) = \mu_{\boxplus}(\mathcal{Y}, y_0) \tag{2.33}$$

$$\mu_{\boxplus}(\mathcal{Y}, \mu) = \begin{cases} \mu \boxplus e & \|e\| < \epsilon \\ \mu_{\boxplus}(\mathcal{Y}, \mu \boxplus e) & \text{otherwise} \end{cases} \quad \text{with } e = \frac{1}{2n+1} \sum_{i=0}^{2n} y_i \boxminus \mu \quad . \tag{2.34}$$

The covariance of two sets of manifold element sigma points, $\mathcal{X}$ and $\mathcal{Y}$, each with $(2n + 1)$ elements, is

$$\Sigma_{\boxplus}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \sum_{i=0}^{2n} (x_i \boxminus \mu_{\boxplus}(\mathcal{X})) (y_i \boxminus \mu_{\boxplus}(\mathcal{Y})) \quad . \tag{2.35}$$

The propagation of the sigma points through the dynamic and measurement model functions is unchanged. The functions are problem specific, and thus know how to deal with their problem specific manifold elements.

UKF on $\boxplus$-manifolds The UKF on $\boxplus$-manifolds is a generalization of the ordinary UKF in that the state, dynamic input and measurement are $\boxplus$-manifold elements, $x^{(t)} \in \mathbb{M}_x, u^{(t)} \in \mathbb{M}_u, z^{(t)} \in \mathbb{M}_z$. The system dynamics function, $f : \mathbb{M}_x \times \mathbb{M}_u \to \mathbb{M}_x$, and the measurement model function, $h : \mathbb{M}_x \to \mathbb{M}_z$, correspondingly also work on elements of $\boxplus$-manifolds.

The dynamic update of the UKF on $\boxplus$-manifolds to obtain the intermediate mean and covariance is as follows. Equation (2.32) is used to generate manifold-element sigma points from the previous mean and covariance as well as the dynamic input and its covariance. The sigma points are then propagated through the dynamic model function to obtain the intermediate mean and covariance,

as they were in the ordinary UKF in eq. (2.14):

$$\bar{\mathcal{X}}^{(t)} = f(\mathcal{X}^{(t)}) \quad \text{with } \mathcal{X}^{(t)} = \text{sp}_{\boxplus}\left(\begin{bmatrix}\hat{x}^{(t-1)} \\ u^{(t)}\end{bmatrix}, \begin{bmatrix}\Sigma^{(t-1)} & 0 \\ 0 & \Sigma_u^{(t)}\end{bmatrix}\right) \tag{2.36}$$

$$\bar{x}^{(t)} = \mu_{\boxplus}\left(\bar{\mathcal{X}}^{(t)}\right) \tag{2.37}$$

$$\bar{\Sigma}^{(t)} = \Sigma_{\boxplus}\left(\bar{\mathcal{X}}^{(t)}, \bar{\mathcal{X}}^{(t)}\right) \tag{2.38}$$

Again, as in the ordinary UKF, the intermediate sigma points are propagated through the measurement model function. The results of the measurement function are manifold elements themselves,

$$\mathcal{Z}^{(t)} = h\left(\bar{\mathcal{X}}^{(t)}\right) \ . \tag{2.39}$$

From this, the updated estimate is computed as follows.

$$K^{(t)} = \Sigma_{\boxplus}\left(\mathcal{Z}^{(t)}, \bar{\mathcal{X}}^{(t)}\right)^T \left[\Sigma_{\boxplus}\left(\mathcal{Z}^{(t)}, \mathcal{Z}^{(t)}\right) + \Sigma_z^{(t)}\right]^{-1} \tag{2.40}$$

$$\tilde{\Sigma}^{(t)} = \bar{\Sigma}^{(t)} - K^{(t)}\Sigma\left(\mathcal{Z}^{(t)}, \bar{\mathcal{X}}^{(t)}\right) \tag{2.41}$$

$$\delta = K^{(t)}\left(z^{(t)} \boxminus \mu_{\boxplus}\left(\mathcal{Z}^{(t)}\right)\right) \tag{2.42}$$

$$\tilde{\mathcal{X}}^{(t)} = \{\bar{x}^{(t)} \boxplus d | d \in \Delta\} \quad \text{with } \Delta = \text{sp}\left(\delta, \tilde{\Sigma}^{(t)}\right) \tag{2.43}$$

$$\hat{x}^{(t)} = \mu_{\boxplus}\left(\tilde{\mathcal{X}}^{(t)}\right) \tag{2.44}$$

$$\Sigma^{(t)} = \Sigma_{\boxplus}\left(\tilde{\mathcal{X}}^{(t)}, \tilde{\mathcal{X}}^{(t)}\right) \tag{2.45}$$

There is an additional sigma point propagation to compute $\hat{x}^{(t)}$ and $\Sigma^{(t)}$, which is not present in the original UKF. It is necessary to convert the covariance from the old reference point to the new one, i. e. to get to

$$\hat{x}^{(t)} \boxplus \mathcal{N}\left(0, \Sigma^{(t)}\right) \text{ from } \bar{x}^{(t)} \boxplus \mathcal{N}\left(\delta, \tilde{\Sigma}^{(t)}\right) \ . \tag{2.46}$$

Orientation-UKF Hertzberg et al. happened to pick pose estimation using GPS and INS as an example application for the $\boxplus$-UKF [Her+13]. Orientation estimation using an inertial sensor and a magnetometer is very similar. The manifold of the state is $\mathbb{SO}(3)$, with the following implementations of $\boxplus : \mathbb{SO}(3) \times \mathbb{R}^3 \to \mathbb{SO}(3)$ and $\boxminus : \mathbb{SO}(3) \times \mathbb{SO}(3) \to \mathbb{R}^3$

$$Q_2 = Q_1 \boxplus q = Q_1 \,\text{Rot}(q) \quad Q_2 \boxminus Q_1 = \text{aRot}(Q_1^T Q_2) \ , \tag{2.47}$$

with $\text{Rot}(q) = \exp(q_\times)$ and its inverse $q = \text{aRot}(\text{Rot}(q))$. $q_\times$ is the matrix corresponding to the cross-product of q with any vector $v \in \mathbb{R}^3$, such that $q_\times v = q \times v$. aRot is essentially the matrix logarithm, but returns the vector q instead of the cross-product matrix $q_\times = \log(\exp(q_\times))$. The operators imply rotation matrices as the representation of the $\mathbb{SO}(3)$-elements. Other representations, such as unit quaternions, would work just as well.

Defining the dynamic model, $f : \mathbb{SO}(3) \times \mathbb{R}^3 \to \mathbb{SO}(3)$, to integrate the gyroscope measurement $\omega \in \mathbb{R}^3$ over the sampling time δt and the measurement model, $h : \mathbb{SO}(3) \to \mathbb{R}^6$, to predict the accelerometer and magnetometer measurements $\begin{bmatrix} z_g^T & z_m^T \end{bmatrix}^T = z \in \mathbb{R}^6$ from gravity g, and the earth's magnetic field m, completes the simple estimator:

$$f(Q, \omega) = Q \operatorname{Rot}(\omega \delta t) \quad h(Q) = \begin{bmatrix} -Q^T g \\ Q^T m \end{bmatrix} \quad . \tag{2.48}$$

This is not exactly the same as Kraft's orientation estimator; the angular velocity is not part of the state. However, it uses the tricks used by Kraft [Kra03] to make the UKF cope with orientations, but neatly packaged into the $\boxplus$-operators, making it very easy to apply them to other $\boxplus$-manifolds. Examples are given in [Her+13], along with juxtapositions of UKF and Gauss-Newton algorithms with their corresponding $\boxplus$-versions.

Discussion

There are even more estimators to mention. There is another class of estimation algorithms, called particle filters. A particle filters is a discretization of the Bayes Filter, which does not require the distribution of the estimated quantity to be Gaussian [TBF05]. Since the distribution is represented by discrete particles, the number of particles to represent a high-dimensional state, e. g. the posture of a human skeleton, becomes very big. Suffering from the "curse of dimensionality" [TBF05] so badly, they are not applicable for this work.

The main problems of orientation estimation are non-linearity and singularities. In approaches which estimate an over-parameterized orientation representation and enforce its structure using constraints, the non-linearity becomes an even bigger issue, because the constraints are non-linear. There are more algorithms with an emphasis on linearization, which have not been mentioned yet. These include the Two-Step Attitude Estimator and the Predictive Filter [CMC07]. The latter's main advantage is that it linearizes at the output and not at the system dynamics and thus avoids applying a structure preserving constraint in the system dynamics. The former looks like an EKF, but it establishes an intermediate state representation, the so called first-step state, in which an otherwise non-linear measurement model becomes linear.

Specialized orientation estimators, which know about the structure, appear to be superior. They are, however, specialized to orientations. Thus, if elements of another ($\boxplus$-)manifold are to be estimated, another specialization has to be developed. The need to estimate other elements of other manifolds will come up in this work. Against this background, an encapsulation of the peculiarities of each manifold in the $\boxplus/\boxminus$-operations, making generic algorithms applicable almost immediately, is very attractive.

The following strategy to develop new estimators of quantities including manifold elements appears to be most practical: Use a generic estimator with $\boxplus/\boxminus$ operators, whose initial state is computed using a specialized estimator. That way, one gets the great convergence when little of the state is known and a reasonably simple estimator using a well-understood, generic algo-

rithm. Additionally, one neither has to worry about singularities, i. e. no need for "tricks" as in the MEKF, nor are there any non-linear structure-preserving constraints to apply.

2.2 Posture and Joint Angle Estimation

Many approaches have been developed to estimate joint angles or skeleton postures using inertial sensors. Some consider the human skeleton as a whole, others parts of it such as arms or lower limbs only. Some use magnetometers, some try to estimate postures without them.

Luinge [Lui02; LV05] found that, without additional means, Kalman-Filtering body segment orientations using gyroscope and accelerometer measurements does not yield better heading estimates than simple integration of gyroscope data. This is no surprise, because there is no source of absolute heading information if the skeleton structure is not used.

Joint Angles

Cheng et al. [CO10] surveyed techniques to estimate joint angles with accelerometers and gyroscopes only, by using additional means, namely kinematic properties of a skeleton of rigid bodies. However, they restrict themselves to biaxial accelerometers and uniaxial gyroscopes, so the techniques surveyed are applicable only to hinges, e. g. elbows, and not to spherical joints, e. g. shoulders.

All techniques surveyed by Cheng et al. use that a joint angle can be computed from the difference of the linear accelerations of two jointed bodies at the joint center. The techniques differ in how they handle the fact that sensors are not mounted in the joint center and thus linear displacement accelerations measured due to rotation about the joint have to be compensated for.

The surveyed options are: ignoring the problem; differentiating the gyroscope signals to obtain angular accelerations to compute the displacement accelerations; and integrating the gyroscope signal while rotating, using the accelerations only when no rotation is "detected" based on some angular velocity threshold. Alternatively, the use of two displaced accelerometers per body to compute the displacement acceleration is also analyzed.

Being free of thresholds and yielding reasonable performance even if the joint moves, the method involving differentiating the gyroscope signal is recommended by the authors. It is essentially a 2-dimensional version of the idea looked at next and in a little more detail: Seel et al. estimate the angle about an arbitrary, 3-dimensional joint axis with arbitrarily aligned sensors.

Similarly to this work, Seel et al. use kinematic constraints to calibrate the pose of inertial sensors relative to their respective bodies and to estimate the angle between two jointed bodies for the purpose of gait analysis [SSR12; SRS14]. They also exploit the "obvious" [Liu+09] fact, that the linear acceleration at the joint location can be measured with inertial sensors on either side of the joint. That this can be used to obtain joint angles has been demonstrated by Liu

et al. back in 2009 [Liu+09]. Let $a_{J,S}$ be the linear acceleration at the location of joint J in the reference frame of sensor S. Further, let a_S be the linear acceleration at the location of sensor S. On a rigid body, which does not rotate, we obviously have $a_{J,S} = a_S$. On a rotating body $a_{J,S} = a_S + \Gamma(\omega_S, \dot{\omega}_S, r_{J,S})$, where $\Gamma(\omega, \dot{\omega}, r)$ represents the radial, centripetal and Coriolis accelerations introduced by changing the reference point of the acceleration to $r_{J,S}$, the location of the joint in sensor coordinates. Of course, the accelerations introduced by the rotation also depend on the angular velocity and angular acceleration of the sensor S, ω_S and $\dot{\omega}_S$.

For two sensors $S \in \{1, 2\}$ mounted on bodies, connected over the joint J, $a_{J,1}$ and $a_{J,2}$ must be the same up to sensor noise and the relative rotation of the two sensors, that is $a_{J,2} = Q_{2\curvearrowleft 1} a_{J,1}$.

As Seel et al. are concerned with hinges, i. e. joints with a dominant joint axis, they make no attempt to estimate the complete relative orientation matrix $Q_{2\curvearrowleft 1}$. Instead, they project $a_{J,1}$ and $a_{J,2}$ into the plane perpendicular to the joint axis and compute the joint angle α_J between the two projections. The angular velocity of the joint is easily computed from the two angular velocity measurements. Let the joint axis be $j_{J,S}$ in coordinates of sensor S, then $\dot{\alpha}_J = j_{J,2}^T \omega_2 - j_{J,1}^T \omega_1$ [SRS14].

With the relative $\dot{\alpha}$ and the absolute α, there are all ingredients necessary for any predictor-corrector estimator, e. g. a Kalman Filter, to estimate the joint angle over time.

Since Seel at al. are concerned with recovering the joint angle and not with estimating the complete posture, they can discard all information about the sensor orientations in a world-fixed reference frame. Of course, it is impossible to recover the global heading without a magnetometer. However, while it is not an issue for gait analysis, a posture estimator should maintain where up and down are in a global sense: It clearly is a difference whether someone is holding his hands up or is doing a handstand, although the joint angles might be the same. The proposed method also requires numerically differentiating the noisy gyroscope signal to obtain $\dot{\omega}$.

Seel et al. are not only concerned with the joint angle estimation problem. In addition, and that is probably their main contribution, they solved the calibration problem to find the pose of the inertial sensor relative to a body-fixed coordinate system [SSR12]. Using the fact that all $a_{J,S}$ are of the same length (again, up to sensor noise), they compute for sensors $S \in \{1, 2\}$ and joint J with N measurements from each sensor:

$$\arg \min_{r_{J,1}, r_{J,2}} \sum_{k=1}^{N} (\|a_1^{(k)} + \Gamma(\omega_1^{(k)}, \dot{\omega}_1^{(k)}, r_{J,1})\|_2 - \|a_2^{(k)} + \Gamma(\omega_2^{(k)}, \dot{\omega}_2^{(k)}, r_{J,2})\|_2)^2 \ . \tag{2.49}$$

This yields the joint locations in the respective sensor reference frames. Similarly they identify the dominant rotation axis of the joint by observing that the difference of the projections of the angular velocities into the plane perpendicular to the joint axis must vanish. While the latter is only useful for joints that have such an axis, eq. (2.49) is applicable to joints with 3 rotational degrees of freedom. For the skeleton, a combination of joint axis identification and known reference postures or movements, such as in e. g. [Fav+09], appears to be most promising.

For a posture estimator embedded in workwear, such an automatic calibration technique is required, as it is impossible to specify all possible joint locations and sensor orientations, because they do not only depend on the individual suit but also on the person wearing the suit.

Salehi et al. [Sal+15] use Seel's work to determine the locations of the joints in sensor coordinates for multiple bodies at once. Compared to an isolated calibration problem per joint, this leads to a calibrator more robust against bad calibration motions. From the calibrator's perspective, a motion is bad if DOF exist which are moved very little. Their estimator, however, does not estimate the hinge axes or relative orientations of the sensors. For these quantities, they rely on the axes being determined individually and on the orientations being estimated separately.

The work by Seel at al. is also covered in an extensive review of the state of "Wearable Sensing for Solid Biomechanics" [Won+15] by Wong et al. The review treats two aspects of solid biomechanics, namely the Kinetics, which is the identification of forces leading to motions of a rigid body skeleton, and Kinematics, which is the identification of such motions themselves. Wong et al. identify inertial and magnetic sensing as both the least invasive and most portable among the sensing technologies they looked at, which also include mechanical sensors, e. g. exoskeletons like [Bio], and camera-based systems. For the application considered in this dissertation, mechanical approaches such as [Car+14], [Tog+14], although very interesting, are impractical because they would require too much modification of the existing workwear, require direct mounting of the sensors on the limbs and be way to complicated to set up for daily use.

Magnetometers and Disturbance Rejection

Most inertial sensor systems are equipped with complementary sensors, such as a magnetometers or cameras, to compensate for drift around the axis of gravity which occurs when estimating a skeleton's segment orientation. Considering sensors which can be integrated into workwear, the only feasible solution appears to be a magnetometer. Given that the earth magnetic field is almost negligibly small compared to the disturbances encountered on, for instance, a shipyard, those disturbances have to be accurately accounted for.

Roetenberg et al. [Roe+05] appear to be the most successful. Their strategy is to include the magnetic field vector in the world-aligned reference frame into the state estimate. They model the disturbance $d \in \mathbb{R}^3$ on the magnetic field as autocorrelated noise, i. e. the disturbance at time step t is

$$d^{(t)} = d^{(t-1)}c + w^{(t)} \quad , \tag{2.50}$$

where $0 \leq c \leq 1$ and w_t is drawn from zero-mean white noise with covariance $I\sigma_d^2$. With the earth magnetic field vector at time t, $m^{(t)}$, and the sensor orientation estimate $Q_{W\urcorner S}$, the model for the magnetometer measurement at time t is

$$z_m^{(t)} = \left[Q_{W\leftarrow S}^{(t)}\right]^T \left(m^{(t)} + d^{(t)}\right) \quad . \tag{2.51}$$

Given a magnetometer measurement, eq. (2.51) can be solved for $m^{(t)}$ to compute its inclination angle $\phi^{(t)}$ and magnitude. Using those, Roetenberg et al. define the standard deviation of the disturbance to be

$$\sigma_d = \sigma_m |\|m^{(t)}\| - \|m^{(t-1)}\|| + \sigma_\phi |\phi^{(t)} - \phi^{(t-1)}| \tag{2.52}$$

with constant σ_m and σ_ϕ. If the magnetometer measurement changes do not agree with the

orientation estimate, particularly if the magnitude changes, the uncertainty of the disturbance estimate is increased to prevent the magnetometer measurement from deteriorating the orientation estimate.

Equation (2.52) probably works very well when a disturbance appears rather suddenly and then stays approximately constant. However, in situations of frequently changing disturbances, σ_d will be always large, and the magnetometer measurement will not be used to correct the orientation estimate, leading to an approximate no-magnetometer-situation. Consequently, Roetenberg et al. conclude that orientation estimate's accuracy could decrease if the magnetometer is permanently disturbed [Roe+05].

According to Wong et al., there is no effective way to deal with continuously changing interferences over "prolonged time periods" [Won+15]. That this is independent of the algorithm used to estimate orientations is further supported by a comparison [You09] by Young, where the yaw RMS error, i. e. the error about the axis of gravity, is about three times larger than the roll or pitch error, no matter which estimation algorithm was used.

Complete Relative Orientations and Poses

As demonstrated by Seel et al., model constraints improve the situation, because they provide additional information to the estimator, e. g. to make use of the fact that rigid body segments of a skeleton do not move independently. Luinge et al. for instance also recover the exact orientation over an elbow by restricting elbow's degree of freedom [LVB07]. They, however, do not go as far as restricting the elbow to a hinge.

Instead, they estimate, using only gyro- and accelerometers, orientations for the upper and lower arm separately. These are, of course, subject to drift around the world-vertical. In kind of a post-processing step, they compute the relative orientation of the two body segments. The corrected relative orientation then is the — in a least-squares sense — closest orientation which does not rotate around the adduction axis. Since only one degree of freedom is removed, they could not use a hinge rotation axis to define the body segment coordinate systems. Predefined motions and thus approximately known accelerations and angular velocities along and around the axes of the body segments' coordinate systems were used. This caused problems in the orientation correction, i. e. orientation errors exceeding 20°, because the adduction constraint was defined in physical body segment coordinates and the bearer of the sensors, of course, could not execute the predefined calibration motion with perfect accuracy.

This suggests that the references for constraints should be, whenever possible, specified in the frames of reference of the sensors, avoiding systematic inaccuracies in the constraints an estimator uses.

The presented works so far, that do without complementary sensors, restrict the rotary degrees of freedom of joints and do not consider spherical joints.

Kok et al. [KHS14] experimentally showed that postures can be estimated without restricting the rotary degrees of freedom, i. e. with spherical joints. They stated the posture estimation problem as an offline optimization problem, where the essential constraint is that two adjacent

body segments are inseparably connected by a joint. Doing straightforward INS-like integration of angular velocity and double-integration of acceleration, the posture of the complete skeleton is recovered by imposing that the pose of a rigid body is determined by the pose of the predecessor body, if the relative orientation over the connecting joint is known. Conversely, the relative orientation over the joint is the one most likely explaining the inertial measurements.

During patent research in the context of potential commercial applications of the results of this dissertation, patent *US2011028865 (A1)*, which is assigned to Xsens, surfaced. In the patent the authors essentially claim any method to recover the relative orientation of two bodies connected by a joint using accelerometers and gyroscopes mounted on those bodies, exploiting the fact that the joint itself can only have one linear acceleration, which is observable anywhere on the rigid bodies attached to the joint [LRS11]. Although the patent was assigned as early as 2009, it slipped through the patent research for the application of the project SIRKA. The patent, in addition to the claim of the mentioned law of physics, also includes the following algorithm to exploit it.

Similar to the MEKF for orientations, the Inertial Sensor Kinematic Coupling algorithm, as it is called in [LRS11], maintains for each sensor an integrated orientation estimate and an orientation error. The former is obtained by integrating gyroscope measurements. The latter is part of a state vector of a Kalman Filter in conjunction with the position, linear velocity, lowpass-filtered linear acceleration and gyroscope bias of the respective sensor. Although some symbols of the dynamic update description in [LRS11] are unexplained or entirely undefined, it is clear that the dynamic update integrates some sort of gyroscope error to the orientation error, and the measurement of the accelerometer, which is rotated into the global reference frame and corrected for gravity, to the velocity and position. The rotated, gravity-corrected accelerometer measurement is also used for the lowpass-filtered linear acceleration.

The relative position of two sensors on bodies connected over a common joint can be computed using different components of the state: Obviously from the positions but, if the constant joint locations relative to the sensors are known, also from the orientations (i. e. the orientation errors from the state and the separately integrated orientations), by simply computing the difference between the joint locations rotated into global-coordinates. The measurement model then is that the difference between the relative positions obtained both ways must be zero.

There is an additional measurement model saying that the lowpass-filtered acceleration has to align with the gravity vector. The latter model corrects the inclination of the respective sensor, while the former corrects all other state components, including the relative orientations of directly connected bodies, which is the main effect. Finally, the orientation error components of the estimator state are added to the integrated orientations and zeroed in the state vector.

A computational aspect to note about [LRS11] is that the state is pretty high-dimensional. With the positions, their first and second derivative, and orientation errors and gyro biases all being 3-dimensional quantities, there are 15 components per sensor, or at least 12 components, if one removes the measurement update to correct the inclination, which allows to remove the lowpass-filtered orientation from the state.

Also in a cascaded estimator setup, Zihajehzadeh et al. [Zih+15] estimate the orientations of each human limb from the pelvis downwards. The orientations are estimated with an inertial

sensor and a magnetometer per limb. Limbs are concatenated according to the limb lengths. The limb orientations are fused with ultra-wide band measurements in a separate estimator, yielding a posture estimator which includes the global heading and position.

Young [You10] combines per-limb orientation estimation with exploiting the structure of the skeleton. He also uses the skeleton's kinematics to compute the linear accelerations due to rotations of joints.

However, he does not use them to compute the accelerations at a joint center to compute a relative orientation. Instead he subtracts them from the accelerometer measurements to obtain measurements of negative gravity on each limb, even if the joints are moving. i. e. he uses the skeleton structure to reject "gravity disturbances" when estimating orientations of individual limbs individually using gravity and magnetic field.

O'Donovan et al. [ODo+07] found per-segment orientation estimation impractical for joint angle determination. They identified compensating magnetic disturbances in a global reference frame as the main obstacle. Using the usual setup, i. e. one inertial and magnetic sensor on two connected bodies each, they work around the problem by establishing instantaneous reference frames.

To obtain relative orientations, they consider the acceleration and the magnetic field, including disturbances, to be approximately equal at both sensors. From both measurements, the sensor rotation with respect to the instantaneous reference frame defined by magnetic field and acceleration is calculated. The measured accelerations are not compensated for accelerations caused by rotations. If two accelerometer measurements differ in magnitude or if the gyroscope measures a significant rotation, previous accelerometer measurements are rotated into the current sensor coordinates using a rotation matrix obtained from integrating gyroscope measurements.

The alignment of the sensor to its body is pre-determined by rotating the bodies around known axes and observing the gyroscope. No estimator beyond low-pass-filtering the sensor data was used. O'Donovan et al. evaluated their technique by determining joint angles of an ankle during various exercises. Depending on exercise, their approach was between 0.5° and 4° off the estimates of a marker-based camera system.

2.3 Suits

In addition to work on the individual components related to this work, there are complete motion tracking suits.

Salehi et al. developed a motion capturing suit with the sensors integrated into clothes [Sal+13]. They use both inertial sensors and complementary magnetometers, such that they can completely observe the orientations of the bodies equipped with IMUs. Their focus is less on the individual motion capturing techniques than on integrating them into a practical, working system. One step towards this goal is integrating the sensors into tight clothes, which press the sensors, which themselves are mounted on $15mm \times 15mm$ circuit boards, onto their respective limbs. In

combination with the rubber enclosure, they hope to prevent unwanted sensor movements over the skin.

To estimate postures using the inertial sensors, a central computer, called "controlling unit" (CU), reads the inertial sensors and runs the sensor fusion. To achieve high data rates and low energy consumption, they use a wired sensor network. The IMUs are set up in cascaded master-slave-configurations, where the slave IMUs are mounted further down in the kinematic tree, i. e. on the fore arm and upper arm, than the corresponding master IMU, which, in case of arm-slaves, is mounted on the trunk. The slaves write their data into a buffer on the master IMU, which in turn provides all data of its respective kinematic subtree to the sensor fusion.

This cleverly avoids switching over all IMUs for data acquisition. The structure is also mimicked in the sensor fusion algorithm, which runs one EKF per limb to estimate the orientation of the respective body. The EKF corresponding to the master IMU estimates the trunk's orientation relative to a world-fixed reference frame. In addition to the sensor data of the slave-limbs, the trunk's orientation is fed to the EKFs estimating the parameters of the joint connecting to the slave's body segment, e. g. the upper arm. Thus, the master's orientation can be used to exploit constraints on the motion of the joints. Practically this is done by estimating the joint parameters according to a joint model and their first and second derivatives [Mie+13].

Salehi et al. also use an interesting method to calibrate the intrinsic parameters of the inertial sensors. They mount a sensor on one face of a precise cube, which allows to rotate the sensor in steps of exactly 90°. This provides a known upward-direction and a sequence of orientations whose successive differences are all right angles in magnitude.

The pose of the sensors relative to their respective body segments is calibrated from two static postures of the entire skeleton as in [Rei+10]. The positions and segment lengths were not calibrated but simply taken from anthropometric tables.

Salehi et al.'s orientation estimator has a curious feature. Before magnetometer measurements are used in the sensor fusion, they are projected into the ground parallel plane. i. e. the information from the magnetic's field inclination is completely discarded. I suspect that this is done because the magnetometer measurement errors are both correlated and bad enough to corrupt the inclinations estimated from the accelerometers and gyroscopes. I think this is a strong motivation to get rid of magnetometers altogether, because there is little reason to assume that the horizontal component of the magnetometer measurements is any better than the vertical.

The state of the art motion capturing suit is Xsens's MVN [RLS09]. Similar to the SIRKA suit, the MVN contains one IMU per body. Each sensor is contained in a small box, which is strapped on to the corresponding body. The sensors are not integrated into clothing.

How the MVN sensor fusion works is not entirely clear. It is based on Kalman Filtering, where the orientations and positions of individual bodies are integrated as in an ordinary inertial navigation system and are subjected to constraints derived from some biomechanical model. The most import constraint appears to be that the positions of the individual bodies are constrained by joints. It is probably an implementation based on the patent mentioned above. However, the MVN includes magnetometers to correct heading errors [RLS09], although those should not be necessary.

Xsens has competition. Trivisio sells an inertial motion tracking suit [Tri]. To estimate body orientations, they use 15 inertial sensors and magnetometers, which are strapped on to the human whose skeleton is to be tracked, similarly to the Xsens MVN. They note that magnetometers "may be disabled for industrial environments", but do not state how this would affect performance.

More recently, Noitom came up with a suit made of their "Perception Neurons" [Noi]. Those are very tinily packaged IMUs with magnetometers, which are strapped on to the human skeleton's bodies on arbitrary positions. As far as I know, how their IMU and magnetometer data are fused to reconstruct postures is not published. However, users [RDP16] note that the skeleton the Perception Neurons are mounted on is reconstructed. This suggests that there is probably some model the inertial measurements could be conditioned against, which would make the magnetometers optional; but that is speculative.

A different suit related to the assessment of workplace ergonomics is CUELA [EHS09]. CUELA uses both inertial and mechanical sensors to measure the relative orientations of adjacent bodies. It works in conditions unsuitable for magnetometers by measuring the relative angle of two adjacent bodies around the vertical axis using potentiometers, which have to be aligned with the body coordinates. A complicated sensor fusion scheme is thus replaced by additional hardware. The sensors are mounted outside on top of the usual workwear. The system is heavy (3kg).

2.4 Biomechanial Modeling

Biomechanics can not go completely unmentioned when developing a system to estimate skeleton postures. Biomechanical models have been popular to simulate and animate humans for some time, for instance using the work of Monheit and Badler [MB90].

One of the more complicated joints is the shoulder. A strategy to model it, e. g. by Klopvcar et al. [KTL07], is to model the shoulder as a series of connected tiny bone segments and to impose a set of constraints on the joint angles. Following the work by Klopvcar et al., one ends up with a shoulder model with only 3 DOF, which produces shoulder motions that look very natural. Grochow et al. learned the kinematics of an entire, albeit simplified, skeleton, which leads to very naturally looking full-body motions [Gro+04].

To make posture estimating workwear, the advantage of accurate biomechanical models of the skeleton is limited. The error introduced by the fact that the inertial sensors are not mounted directly on the bones is much bigger than the error introduced by an inaccurate model of the bones' connections.

Because human limbs are not rigid, sensors strapped onto the skin can move independently of their corresponding bone. The movements are known as soft-tissue artifacts. The most popular strategy to deal with them is to "drown" them in noise. This can be done either explicitly, as in [KHS14], or implicitly, i. e. by ignoring them, as in, for instance, [Rei+10].

In the SIRKA suit, the sensors are embedded in the cloths themselves, introducing another level of indirection. Since there is, to my knowledge, no model covering the movements of clothing of mul-

tiple differing types of workwear, there is almost no advantage in improving the accuracy above the well-known, robotic kinematic trees known from textbooks such as Featherstone's [Fea08], i. e. connecting rigid bodies by revolute and spherical joints.

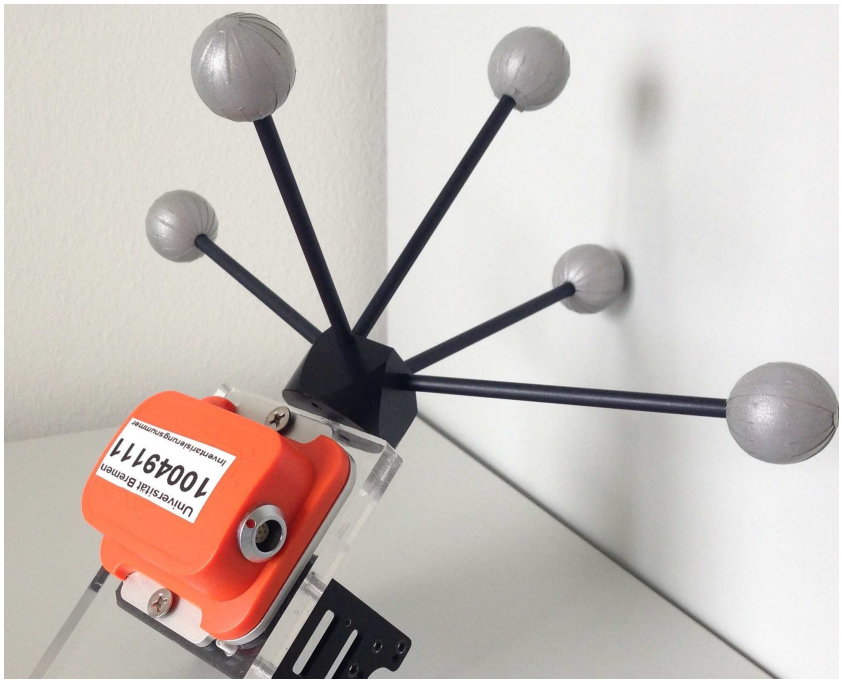


Figure 3.1 *Estimator evaluation device*
An inertial sensor is mounted on a rigid body equipped with camera-trackable markers. The observations of the camera system serve as ground-truth for the estimates inferred from inertial sensor data and prior information.

Chapter 3

Rigid Body Pose Estimation

This chapter is about estimating the pose, mainly the orientation, of a single rigid body. The point of this chapter is not to develop a new algorithm to beat the state of the art in terms of accuracy or speed. It rather aims at a theoretical understanding of how to estimate orientation and position from inertial sensor data. The question here is not so much about practical applications. It rather is about what the assumptions made to estimate an (absolute) orientation from (relative) inertial measurements are. Do alternative assumptions also work and do they tell us something about the position? How can such assumptions be modeled such that a Kalman

Filter can exploit them? These questions are considered by developing three Kalman Filters and testing them on both real and simulated sensor data.

A Kalman Filter is used as a state-of-the-art orientation estimator to obtain parameters of a normal distribution on orientations, a $\boxplus$-manifold. That is, the Kalman Filter estimates $Q \in \mathbb{SO}(3)$ and its covariance $\Sigma \in \mathbb{R}^{3\times3}$, such that an orientation drawn from that distribution is $Q \boxplus \delta q, \delta q \sim \mathcal{N}(0, \Sigma)$ (see Sect. 2.1, p. 18).

A gyroscope measurement $\omega \in \mathbb{R}^3$ over a certain period $\Delta t \in \mathbb{R}$ relates the orientations of two successive time-steps $t-1$ and t to each other, leading to the dynamic model

$$Q^{(t)} = Q^{(t-1)} \operatorname{Rot}(\omega \Delta t). \tag{3.1}$$

To correct the orientation using the gravity measurement, the expected negative-gravity measurement is obtained from the current orientation estimate by $h(Q) = -Q^T g$ and compared to the accelerometer measurement $a_m \in \mathbb{R}^3$.

This is the same as the canonical orientation UKF from Sect. 2.1 (p. 20) without the magnetometer measurement. Without that, it is impossible to observe the heading, but for now that is not the problem studied here.

This approach has a different major issue: The accelerometer measures gravity only if the sensor is not accelerating, which for many, if not most, applications is not the case. How, then, can the negative-gravity measurement model be justified? This is the question to be addressed in this chapter.

3.1 Orientation Estimation with Zero Acceleration Prior

Conceptually, just as in Inertial Navigation Systems (INS), the acceleration is integrated into the state

$$X^{(t)} = \begin{bmatrix} {Q^{(t)}}^T & {x^{(t)}}^T & {v^{(t)}}^T & {a^{(t)}}^T \end{bmatrix}^T . \tag{3.2}$$

The corresponding dynamic model to compute $X^{(t)}$ from $X^{(t-1)}$ using the measured acceleration, a_m, and angular velocity, w, is:

$$X^{(t)} = \begin{bmatrix} Q^{(t-1)} \operatorname{Rot}(\omega \Delta t) \\ x^{(t-1)} + v^{(t-1)} \Delta t + \frac{(\Delta t)^2}{2} \left(Q^{(t-1)} a_m + g \right) \\ v^{(t-1)} + \left(Q^{(t-1)} a_m + g \right) \Delta t \\ Q^{(t-1)} a_m + g \end{bmatrix} . \tag{3.3}$$

In eq. (3.3), the acceleration measurement a_m is rotated to world coordinates such that velocity v and position x are always integrated in the same frame of reference. While this would be unnecessary if x and v were excluded from the state, those quantities will turn out to be useful later in this chapter.

A particular feature of the state in eq. (3.2) and eq. (3.3) is that it contains both ordinary 3-dimensional vectors and the orientation manifold element $Q \in \mathbb{SO}(3)$. This avoids singularities of 3-dimensional orientation parameterizations, but also requires operators to modify the orientation for the Kalman Filter, which does not know about the 3-dimensional structure of the rotation matrix per se. The $\boxplus$ and $\boxminus$ operators for orientations, introduced in eq. (2.47) (Sect. 2.1, p. 20), are used to apply small, vectorially represented changes to the orientation:

$$Q_1 \boxplus \delta q = Q_1 \operatorname{Rot}(\delta q), \quad Q_2 \boxminus Q_1 = \operatorname{aRot}\left(Q_1^T Q_2\right) \qquad \text{(2.47 revisited)}$$

$\boxplus$ and $\boxminus$ are extended to the entire state using eq. (2.47) for the rotation matrix and ordinary vectorial $+$ and $-$ for the remaining vector components:

$$X_1 \boxplus \begin{bmatrix} \delta q \\ \delta x \\ \delta v \\ \delta a \end{bmatrix} = \begin{bmatrix} Q_1 \boxplus \delta q \\ x_1 + \delta x \\ v_1 + \delta v \\ a_1 + \delta a \end{bmatrix}, \quad X_2 \boxminus X_1 = \begin{bmatrix} Q_2 \boxminus Q_1 \\ x_2 - x_1 \\ v_2 - v_1 \\ a_2 - a_1 \end{bmatrix} \tag{3.4}$$

Now in absence of a measurement update, the filter only integrates and never corrects gyro drift. What could be the measurement model now?

Naturally, things we attach sensors to move with a bounded velocity. So it is known that the sensor's acceleration is $a_P = 0$ *on long term average* but with considerable covariance $\Sigma_P \in \mathbb{R}^{3\times3}$.

This knowledge is built into the estimator explicitly. It is formulated as a prior on $a^{(t)}$ for all t:

$$h(X^{(t)}) = a^{(t)} = a_P + \delta^{(t)}, \quad \delta^{(t)} \sim \mathcal{N}(0, \Sigma_P) \quad \forall t \tag{3.5}$$

This prior, or linear soft constraint [Sim10], replaces the measurement model as in [BKL02, chapter 5], with $a_P = 0$ as measurement and Σ_P as measurement covariance. The measurement matrix H is

$$H = \frac{\partial}{\partial X} h(X)\bigg|_{X=\bar{X}^{(t)}} = \frac{\partial}{\partial \delta x} h(\bar{X}^{(t)} \boxplus \delta x)\bigg|_{\delta x=0} = \begin{bmatrix} 0 & 0 & 0 & I_3 \end{bmatrix} . \tag{3.6}$$

It is obtained by differentiating h with respect to every degree of freedom of the state. Because h just picks the acceleration component of the state, it is obviously linear. I_3 is the (3×3) identity matrix.

Thus, the measurement update is

$$K = \bar{\Sigma}^{(t)} H^T (H \bar{\Sigma}^{(t)} H^T + \Sigma_P)^{-1} \tag{3.7}$$

$$\hat{X}^{(t)} = \bar{X}^{(t)}_{|a_P=0} = \bar{X}^{(t)} \boxplus K(0 - h(\bar{X}^{(t)})) . \tag{3.8}$$

Since $\boxplus$ is non-linear in the orientation component of the state, eq. (3.8) and the corresponding covariance $\hat{\Sigma}^{(t)} = \bar{\Sigma}^{(t)}_{|a_P=0}$ are computed by a Sigma Point Propagation through $\boxplus$, as described in Sect. 2.1 and originally in [Her+13]. As an approximation one could also compute (3.8) directly

and update the covariance as in eq. (3.9).

$$\hat{\Sigma}^{(t)} = \bar{\Sigma}^{(t)}_{|a_P=0} = \bar{\Sigma}^{(t)} - KH\bar{\Sigma}^{(t)} \tag{3.9}$$

To summarize, we have a canonical filter (with a generic treatment of rotations) using the canonical dynamic model for an IMU plus a zero acceleration prior.

The question that now arises is, whether the incorporated prior actually provides a correction of the drifting orientation as is the case in the traditional orientation estimator.

This is indeed the case. Even more, as is derived in Sect. 3.9, the first order approximations of an estimator using the acceleration prior and the original filter, which uses the negative-gravity measurement model, are equivalent with respect to the orientation. The only difference is that the covariances of the accelerometer measurements and the zero-acceleration prior are stated separately and are effectively added in the filter.

There is still one problem to address which becomes apparent due to the separation of sensor noise and "noise" in the prior: Built into the Kalman Filter is the requirement that the noise on a measurement is independent from the noise on other measurements. If the prior was applied at every time-step, the "noise on the prior" would most certainly not be independent. For instance, let the measurement frequency be 100Hz, then the accelerations of a human hand at two successive measurements clearly are correlated. To model this, I assume the sequence of accelerations to have at most an exponentially decaying autocovariance. Two accelerations $a_m^{(i)}$ and $a_m^{(j)}$ appearing at points in time $t^{(i)}$ and $t^{(j)}$ in the acceleration series have covariance

$$\left\| \mathrm{Cov}\left(a_m^{(i)}, a_m^{(j)}\right) \right\| \leq \sigma_\rho^2 \exp\left(-\frac{\delta t}{\tau}\|i-j\|\right). \tag{3.10}$$

In eq. (3.10), δt is the time between two successive measurements, σ_ρ the standard deviation and τ the decorrelation time. The latter two depend on the quantity being measured and the environment the sensor is used in. At every time-step, the estimate is conditioned on the prior with covariance

$$\Sigma_\rho = I\sigma_\rho^2\left(1 + 2\frac{\tau}{\delta t}\right) \quad . \tag{3.11}$$

This results in a conservative white-noise approximation of the covariance of the series, i.e., according to the Theorem 1 in Sect. 3.10, for any series $u \in \mathbb{R}^{3n}$ of n-many 3-dimensional accelerations

$$u^T \mathrm{Cov}(a_m^{(1:n)})u \leq u^T u\sigma_\rho^2\left(1 + 2\frac{\tau}{\delta t}\right) \tag{3.12}$$

with $\mathrm{Cov}\left(a_m^{(1:n)}\right)_{i,j} = \mathrm{Cov}\left(a_m^{(i)}, a_m^{(j)}\right)$.

This also explains why the noise on the negative-gravity measurement is traditionally set very large. First, because the correlated prior has a large variance σ_ρ^2 and second due to the additional factor $1 + 2\frac{\tau}{\delta t}$ for the correlation.

3.2 Estimator variations

So the assumption that the sensor is *on average* not accelerating justifies the original orientation estimator. It raises the question though, whether there are alternative valid assumptions as to the translational motion of the sensor, which may yield reasonable orientation estimators as well.

There are two obvious alternatives to the Acceleration-Prior-Filter, i. e. to stating that the average acceleration is zero:

- The velocity $v_p \in \mathbb{R}^3$ is zero on average with large variance, leading to the Velocity-Prior-Filter. This is physically justified for many scenarios, e. g. moving indoors.
- The position $x_p \in \mathbb{R}^3$ is zero on average with large variance, leading to the Position-Prior-Filter. This is also physically justified if the sensor stays in a room.

The corresponding estimators use the same state and dynamic model as the Acceleration-Prior-Filter, but use different priors as measurement model replacements.

The Velocity-Prior-Filter conditions the estimate, analogously to eqs. (3.8) and (3.9), on the velocity prior $v_P + \delta^{(t)} = v^{(t)} = h(X^{(t)})$ with $\frac{\partial h}{\partial X} = H = \begin{bmatrix} 0 & 0 & I_3 & 0 \end{bmatrix}$

$$\hat{X}^{(t)} = \bar{X}^{(t)}_{|v_P=0} \quad \text{and} \quad \hat{\Sigma}^{(t)} = \bar{\Sigma}^{(t)}_{|v_P=0},$$

while the Position-Prior-Filter conditions the estimate on the position $x_P + \delta^{(t)} = x^{(t)} = h(X^{(t)})$ with $\frac{\partial h}{\partial X} = H = \begin{bmatrix} 0 & I_3 & 0 & 0 \end{bmatrix}$

$$\hat{X}^{(t)} = \bar{X}^{(t)}_{|x_P=0} \quad \text{and} \quad \hat{\Sigma}^{(t)} = \bar{\Sigma}^{(t)}_{|x_P=0}.$$

Again, the position component is not necessary for the Velocity-Prior-Filter, but it will be useful in the coming sections and does not affect the orientation estimate, because the velocity prior is linked to the velocity component only by the measurement matrix H.

3.3 Orientation Experiments

To test whether or not the assumptions on the velocity and position yield reasonable orientation estimators, I implemented both as well as the classical orientation estimator as Unscented Kalman Filters (UKFs) [JU97], using the different priors as measurement update replacements.

Since gravity is always vertical, the orientation around the vertical axis, the heading, is unobservable using an accelerometer only. If no other information is added, this causes the corresponding covariance component to grow. That, in turn, causes problems with the representation of orientations in the sigma point propagation, namely the sigma points corresponding to the heading to wrap around. To prevent this, the UKF implementation uses another prior on the y-component

of the IMU's forward (x-)axis, i. e. the heading,

$$h_\alpha(X) = \begin{bmatrix}0 & 1 & 0\end{bmatrix} Q \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T, \tag{3.13}$$

which does not modify the estimate, i.e. the innovation is always zero, but limits the covariance in the heading dimension.

To evaluate the performance of each filter, its orientation estimate, Q, is compared to a ground-truth orientation, $Q_{W\text{ɬgt}} \in \mathbb{SO}(3)$. The latter is provided by bundling an IMU[1] rigidly with a commercial off-the-shelf tracking system[2], whose relative pose was automatically calibrated by predicting the angular velocities and linear accelerations of the IMU from the bundle trajectory observed by the tracker. The bundle is displayed in Fig. 3.1.

Because heading is not observed, the contribution of the heading to the orientation error is not taken into account. Accordingly, the orientation error is the angle between the world-vertical unit vector transformed into IMU coordinates using the orientation estimate and ground-truth respectively. That is

$$\angle\left(Q_{W\text{ɬgt}}^T \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T, Q^T \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T\right) = \arccos\left(\left[Q_{W\text{ɬgt}} Q^T\right]_{3,3}\right). \tag{3.14}$$

As plotted in Fig. 3.2, the differences in orientation estimation performance are very small, so all priors apparently lead to valid estimators, but it is impossible to pick a particular winner.

The plot in Fig. 3.2 is a little unfortunate for another reason. The estimators as well as the integrator were initialized with the ground-truth orientation. The gyroscope is good enough that the inclination error obtained only by integrating is only 1.2° after a minute, so there is very little error for the estimators to correct using the priors. With the initial orientation 2° off of ground truth, the correction effect is much bigger, as plotted in Fig. 3.3.

3.4 Pose Estimation

Aside from the insight that there's more than one assumption to build an orientation estimator on, there might be some merit in trying different priors.

There are numerous applications which require not only a 3-DOF orientation but a complete 6-DOF pose, including the position of a rigid body. Many of such applications do not require a globally correct position, in the sense of locating the object on earth. A local position, revealing where an object is relative to where it was, for instance, 5 seconds ago, is sufficient, even if it remains unknown where both positions are in a global sense. For instance, a head-mounted device in an virtual reality application may be used to track the complete pose of the head. There, the head stays on average at some reference position, and while correct short-term local

[1] MTi from XSens
[2] ARTtrack/DTrack2 from A.R.T. GmbH

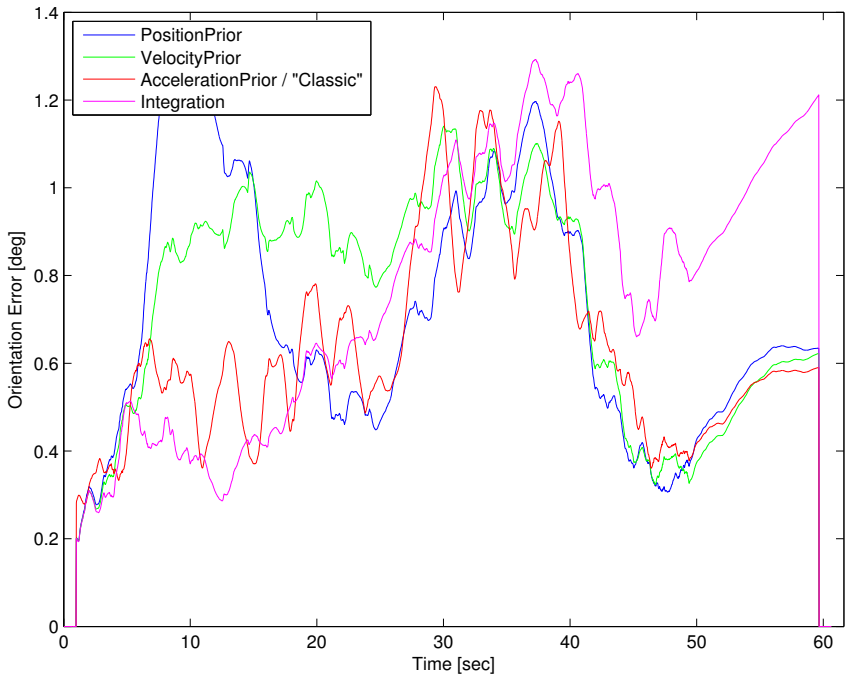


Figure 3.2 *Orientation Errors with Priors*
Orientation errors of each orientation estimator w.r.t. ground-truth. For each estimator, the angle in radians of the relative rotation between estimate and ground-truth is plotted. The rapidly varying error is smoothed by averaging over 0.2s. For comparison, the magenta line shows the average orientation error from uncorrected integration of gyroscope measurements.

poses relative to the average zero-position are critical, its not important where the reference position is globally.

If the initial velocity and orientation are known, local pose estimates are obtainable from inertial measurements by integrating gyroscope and accelerometer measurements over time, which is the basic idea behind inertial navigation systems [BKL02, chapter 12.3]. Due to accumulating sensor noise, the pose estimate deteriorates with time, so if position estimates are to be provided over an extended period of time, some mechanism to reset the position and velocity to known values is necessary. The estimators using prior information and the dynamic model from eq. (3.3) do exactly that integration, but in those estimators position or velocity are also subject to the prior.

To find out whether or not the translation components of the estimator's state have meaningful values, the Velocity-Prior-Filter was exposed to a constant acceleration. Since this acceleration is grossly incompatible with the zero-velocity assumption, the estimated velocity should settle at some constant, non-zero value. As plotted in Fig. 3.4, it did, but the corresponding position estimate is odd.

After a few seconds of believing in a positive velocity, the position should certainly also be positive–and the estimator knows that from the dynamic model. A trajectory as plotted in Fig. 3.4 appears to not make any sense at all. So what's wrong with this estimator?

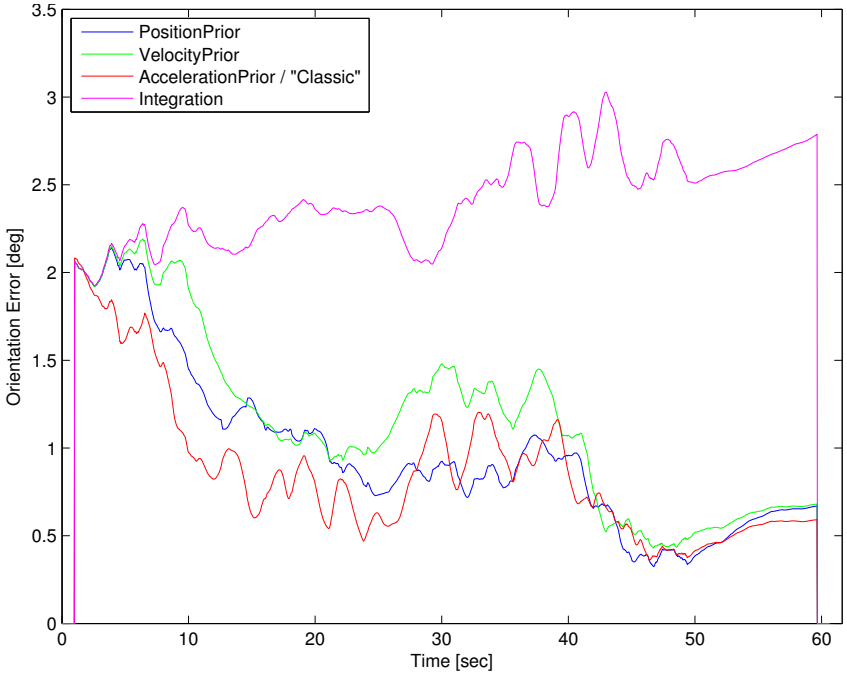


Figure 3.3 *Orientation Errors with Priors (with injected initial error)*
Results of the estimators using the same sensor data as in Fig. 3.2, with an injected initial orientation error of 2°.

To get some insight into this phenomenon, I reduced the state to one velocity dimension and extended it to contain the entire velocity trajectory. That is, in addition to the current velocity the velocities from the past N time steps are also estimated for each time step. Formally the state is $X^{(t')} \in \mathbb{R}^N$ with

$$X^{(t')} = \begin{bmatrix} v^{(t')}_{(t=t')} & v^{(t')}_{(t=t'-1)} & \dots & v^{(t')}_{(t=1)} & 0 & \dots & 0 \end{bmatrix}^T,$$

the constant acceleration is $a \in \mathbb{R}$ and the dynamic model $X^{(t)} = AX^{(t-1)} + Ba$ with $A \in \mathbb{R}^{N \times N}, B \in \mathbb{R}^N$, the $(n-1) \times (n-1)$ identity matrix I and

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ & I & & \vdots \\ & & & 0 \end{bmatrix} \quad B = \begin{bmatrix} \Delta t \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

The velocity assumption is linked to the extended state by $v_P + \delta^{(t)} = v^{(t')}_{(t=t')}$.

The results, plotted in Fig. 3.5, display that the believed past is different from the past belief, which is what is also displayed in Fig. 3.4, with the position being the sum over the believed past. That is, while every time the filter believes it is currently moving, $\forall t' : v^{(t')}_{(t=t')} > 0$, it never

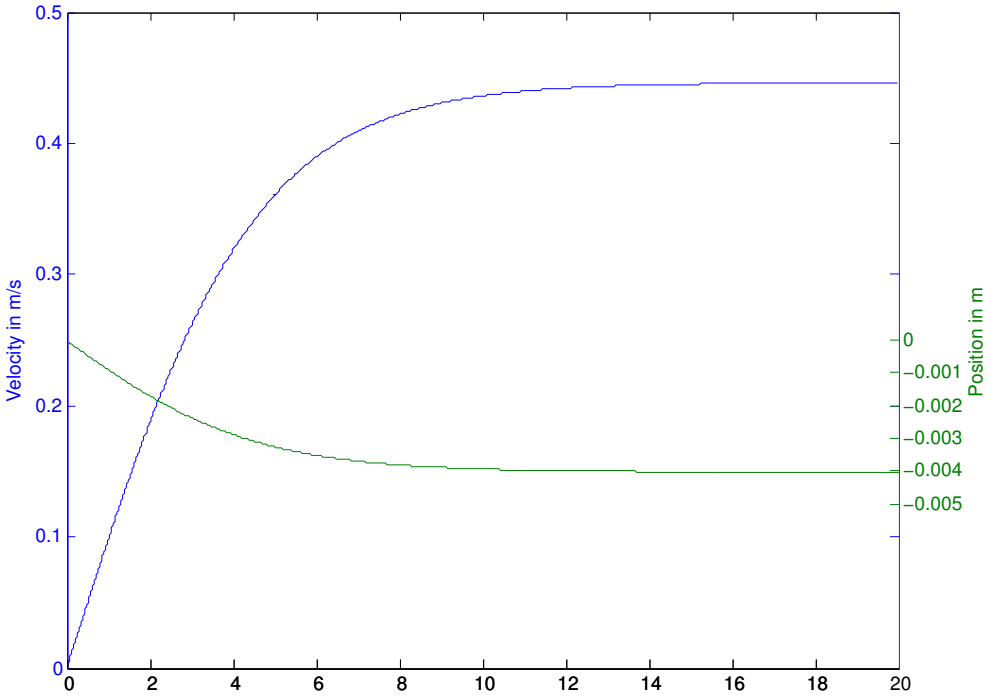


Figure 3.4 *Position-Velocity-Oddity*
Z-components of the velocity and position estimates from a constant acceleration over time, applying the zero-velocity assumption. The position does not increase, although the velocity is positive.

thinks that it has been moving on average in the past, $\Delta t \sum_{t=1}^{t'} v_{(t)}^{(t')} = x^{(t')} \approx 0$.

Thinking again, this behavior makes sense. For a time-step t', the estimator tries to find the trajectory which satisfies best the incompatible assumptions that each velocity $v_{(t)}^{(t')}$ is 0 and that two successive velocities differ by $a\Delta t = v_{(t)}^{(t')} - v_{(t-1)}^{(t')}$. The two ends of a trajectory, best seen in the rightmost trajectory of Fig. 3.5, clearly are compromises. For all t in the middle, $v_{(t)}^{(t')} = 0$ is satisfied completely, which is, although seeming a little one-sided, the best estimate: Improving on one single velocity difference would make all following velocities worse.

This leads to the conclusion that one has to apply a prior to the quantity one is interested in and not to one of its derivatives, i. e. if the complete pose of the IMU is wanted, the Zero-Position-Assumption has to be made. For 6-DOF pose estimation, this rules out all environments in which the zero-position assumption is invalid.

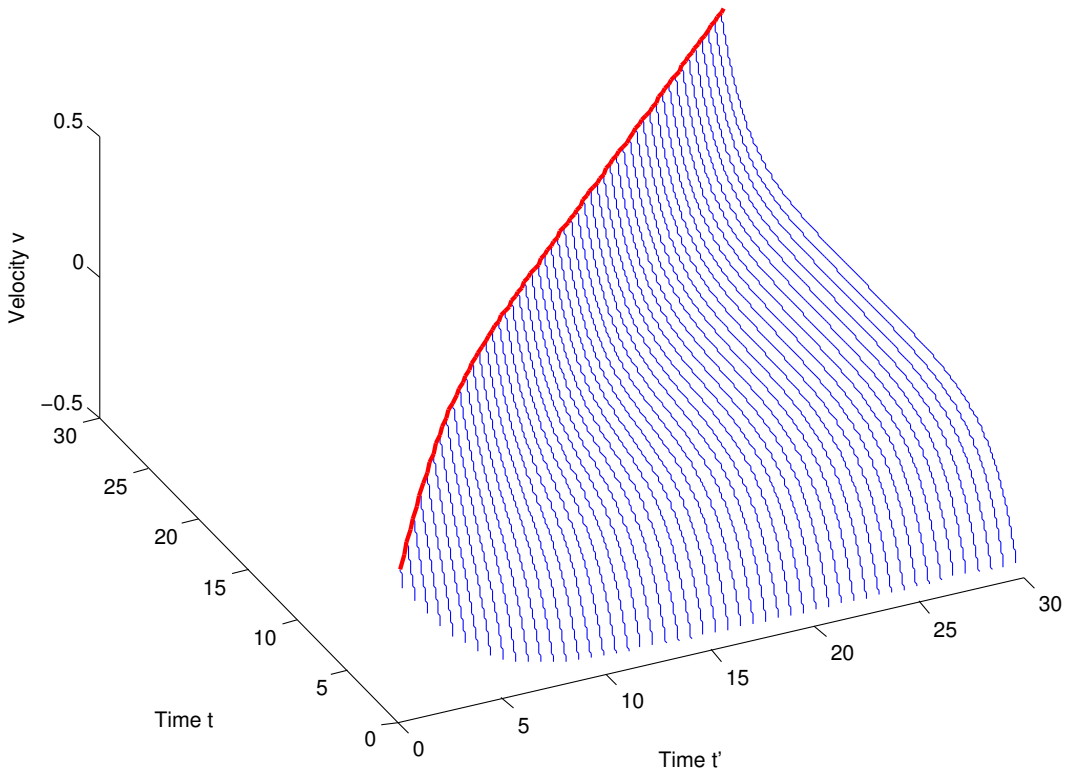


Figure 3.5 *Past belief vs. believed past*
Complete estimated velocity trajectories $v_{(t)}^{(t')}$ (blue), each up to time step t', such that $t \leq t'$. For $t = t'$, the original estimate (red) is reproduced. At time step t', the extended filter believes in past velocities ($t < t'$) different from the past estimates, i. e. the believed past is different from the past belief.

3.5 Translation Experiments

From Sect. 3.3 it is already known that all three estimator variants yield reasonable orientation estimates. From the previous section it is clear that of the three Kalman Filters only the one employing the Zero-Position-Assumption could possibly yield a reasonable position estimate.

For the translational experiments, the same hardware setup as in Sect. 3.3 was used. To evaluate the velocity and – more importantly – the position estimates, their errors with respect to ground truth were compared to the errors of integrating the gravity-corrected accelerometer measurements with respect to ground truth.

What can be expected from such a comparison? Neither the Kalman Filter nor the integrator have access to an absolute position measurement, so the only reasonable thing to ask for is the position of the sensor relative to a starting position from the past. The prior knowledge is that the body and thus the ground truth position does not move far away from the origin. If the starting position lies far in the past, it does not need a Kalman Filter to beat the error of an

acceleration integrator: Integrating accelerations drifts away quickly, and once it drifted away twice the maximum distance of the body from the origin, an "estimator" that always claims that the body is right at the origin would yield a lower error than the acceleration integrator.

It is more interesting to see what happens if the starting position is temporally close. As long as integrated accelerations do not drift further apart than where the body could actually be as modeled by the prior, the estimated position should just follow the doubly integrated acceleration. However, there should be a point where the position drifted so far away from the origin that it becomes gradually implausible given the prior. From then on, the position from the estimator should have a lower error than the position obtained by doubly integrating acceleration.

To test this, the measurement series from Sect. 3.3 was broken into chunks of 5 seconds. At the beginning of each chunk, a conventional orientation estimator with an acceleration integrator was initialized with ground-truth data. Over the remaining 5 seconds, I calculated the velocity error and the error of the position relative to the starting position at the chunk's beginning.

Taking the errors of the relative positions allows to also compare the Position-Prior-Filter to the acceleration integrator when the Position-Prior-Filter is *not* reset to ground truth data at the beginning of each chunk. This obviously puts the Position-Prior-Filter in a disadvantage, but allows to judge its performance in a more practical setting in which the acceleration integrator can not be used due to drift. For a fair comparison, I added second instances of both the Position-Prior- and the Velocity-Prior-Filter, which were reset to ground truth data like the acceleration integrator. The errors are plotted in Fig. 3.6 for the velocity and Fig. 3.7 for the position, respectively. Dashed lines indicate ground-truth resetting.

During the first 1.5 seconds, the Position-Prior-Filter effectively does the same integration as the acceleration integrator, before it gradually starts to correct the position estimate. This is also visible in an accompanying video.[3] Note that after about 3 seconds, the position estimates from both Position-Prior-Filter instances are notably better than the integrated acceleration, although one instance starts off with its own estimate instead of ground truth.

Fig. 3.7 supports the claim that the relative position is locally as accurate as integration but does not drift unboundedly.

3.6 Simulation Model

The estimators using different priors appear to work on actual sensor data. In particular the Position-Prior-Estimator provides estimates superior to integrating the gravity-corrected acceleration. That experiment, though, was rather short-term. There are some properties left for evaluation.

The long term behavior was evaluated using simulated data. It is not entirely clear what a good simulated ground truth would be. To allow the evaluation of statistical properties of the estimator, it has to be generated by random data. The following model was used, which seems

[3] *http://www.informatik.uni-bremen.de/agebv2/downloads/videos/wenk_diss/pose-inertia-prior.mp4*

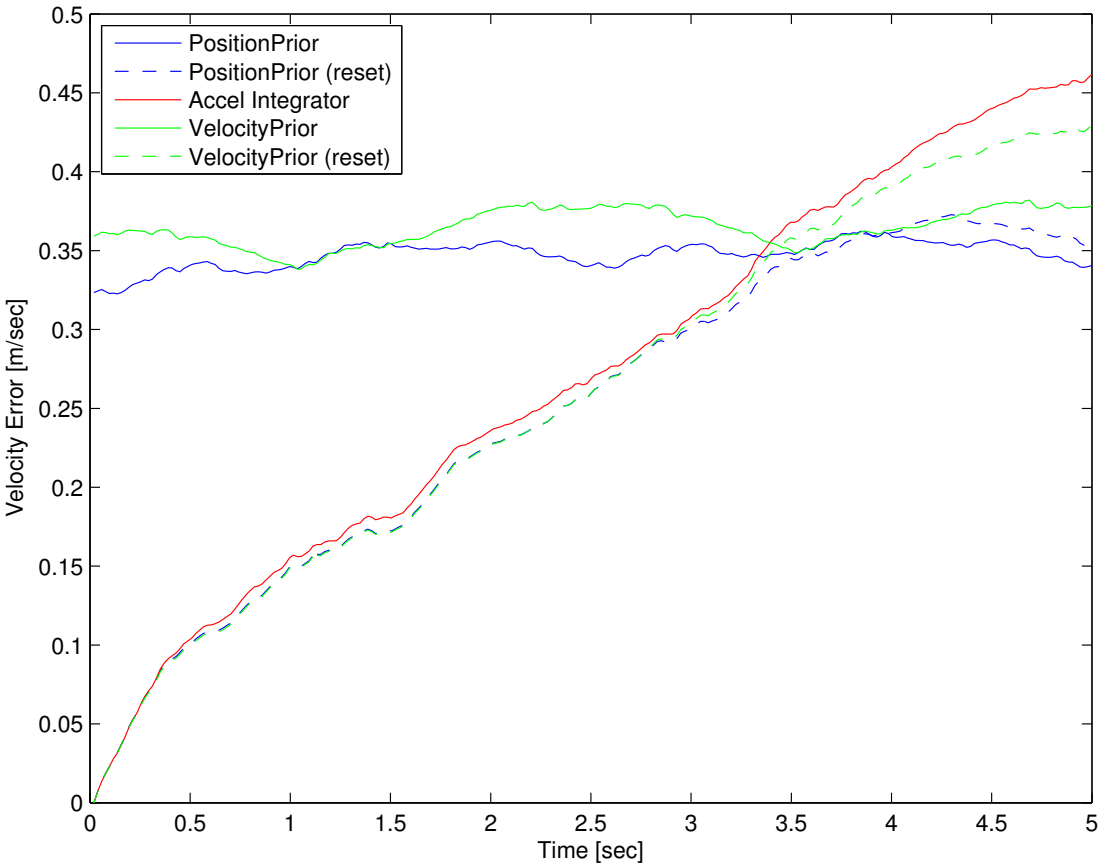


Figure 3.6 *Velocity Estimation Errors*
Average error of the velocity estimates in $[m/s]$, over 5 seconds using the Position-Prior-Filter (blue), the Velocity-Prior-Filter (green) and integrating the gravity-corrected acceleration (red).

to be a reasonable model for movements in a room: It is driven by white noise, the positions are strongly correlated and the variance is limited.

To generate orientations, I sampled $\omega \in \mathbb{R}^3$, $\omega \sim \mathcal{N}(0, I)$ at 100Hz and integrated the samples to orientations.

To generate positions with exponentially decaying autocorrelations, I drove independently for each position component x_ξ, $\xi \in \{x, y, z\}$, a critically damped harmonic oscillator with high-variance white noise accelerations:

$$\ddot{x}_\xi + 2\dot{x}_\xi + x_\xi = z_a \quad \text{with } z_a \sim \mathcal{N}\left(0, \left[10\frac{m}{s^2}\right]^2\right) \tag{3.15}$$

Choosing $\sigma_\rho = 0.56m$ and $\tau = 2.25s$ yields a prior which is satisfied by the generated positions,

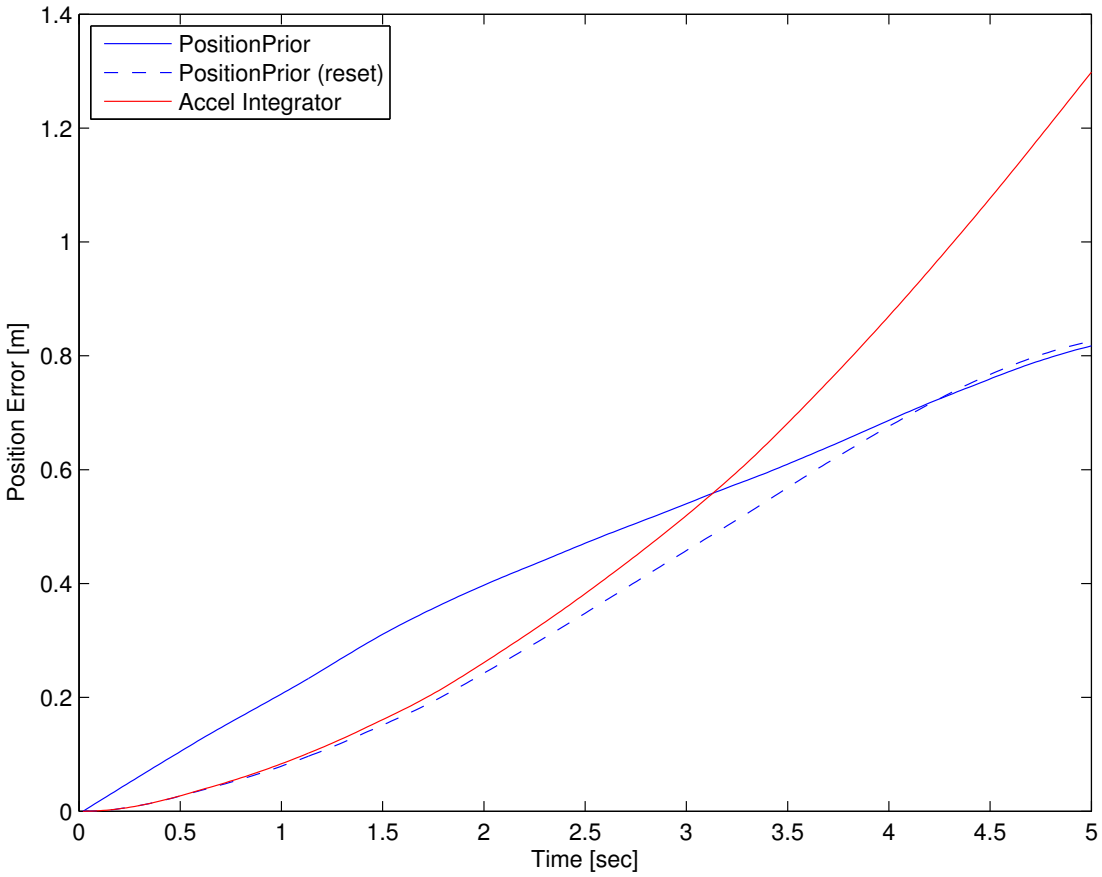


Figure 3.7 *Position Estimation Errors*
Average error of the relative position estimates in $[m]$ over 5 seconds using the Position-Prior-Filter (blue) and integrating the gravity-corrected acceleration (red).

i. e. eq. (3.10) holds for $x^{(i)}$, $x^{(j)}$, so

$$\left\| \mathrm{Cov}\left(x^{(i)}, x^{(j)}\right) \right\| \leq \sigma_\rho^2 \exp\left(-\frac{\delta t}{\tau}|i-j|\right) \quad .$$

Both the autocovariance and the prior covariance are plotted in the top part of Fig. 3.8.

In addition to the positions I took the velocities of the system of eq. (3.15). The lower part of Fig. 3.8 shows the velocities's autocovariance plotted against the velocity prior covariance with $\sigma_\rho = 0.52\frac{m}{s}$ and $\tau = 2s$.

To compute the accelerometer measurements, gravity was subtracted from $\ddot{x}_\xi$. The result was rotated into sensor coordinates. The gyro- and accelerometer measurements were additionally corrupted by white noise generated according to the sensor's specification.

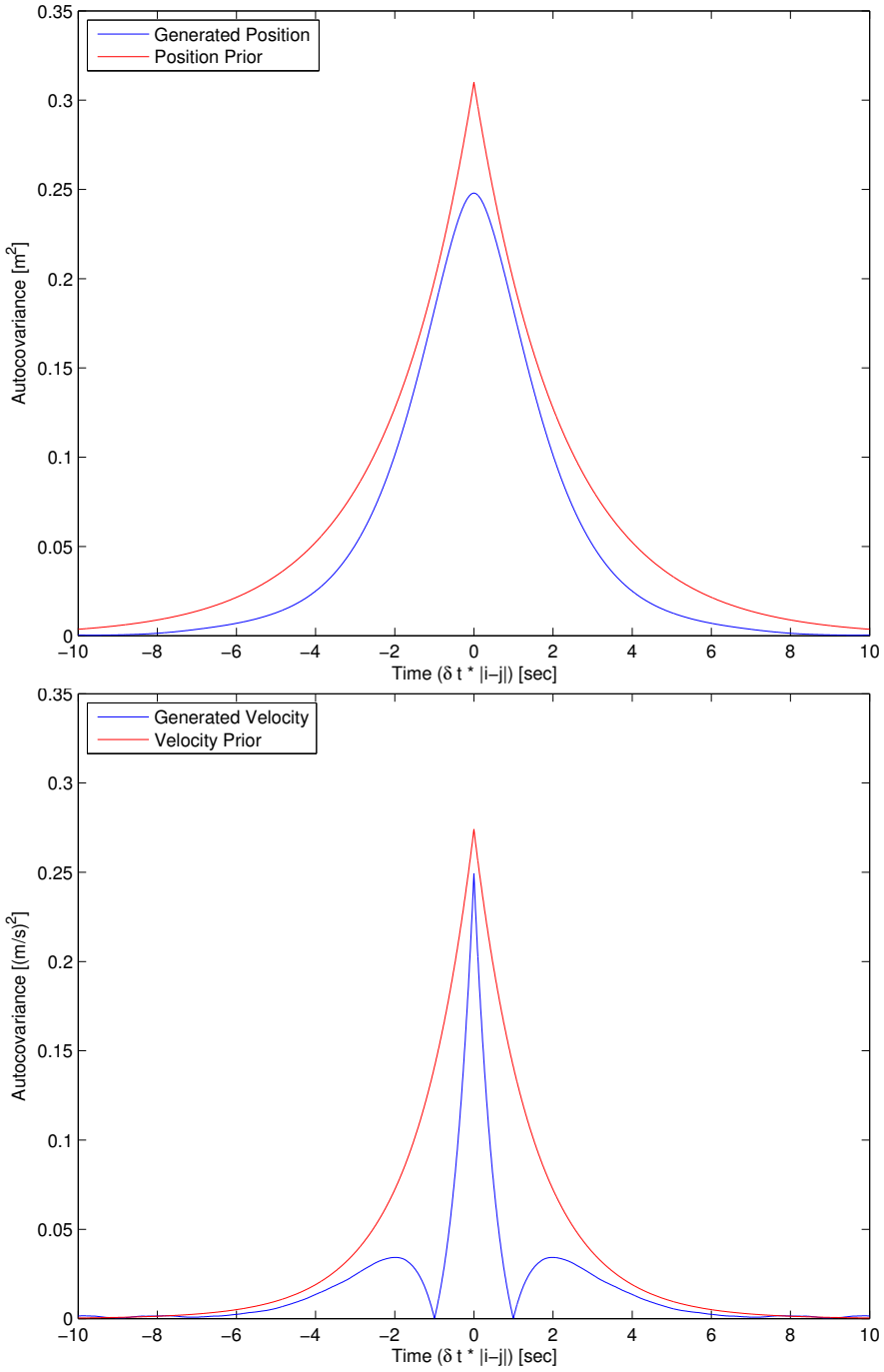


Figure 3.8 *Autocovariance Model*
Absolute autocovariance of the generated sequence of positions (top) and velocities (bottom). With the priors I assume that the absolute autocovariance of the respective sequence (blue) stays below the exponentially decaying autocovariance (red) from eq. (3.10). $\sigma_\rho = 0.56m$ and $\tau = 2.25s$ for the position prior and $\sigma_\rho = 0.52\frac{m}{s}$ and $\tau = 2s$ for the velocity prior were used. The zeros in the velocity's autocorrelation are due to the oscillator character of eq. (3.15) and correspond to a quarter period.

3.7 Prior-Position-Filter Consistency, Boundedness and Long-Term Behavior

To evaluate the consistency of the Prior-Position-Filter, I computed the Normalized Estimation Error Squared (NEES) and the Normalized Mean Estimation Error (NMEE) [BKL02, chapter 5.4] of the individual components from 50 simulations, each 5 minutes long. The estimation error at time step (t) of simulation run (i) is denoted by $\tilde{X}^{(i,t)}$. It is defined as the vectorial $\boxminus$-difference between the quantities estimated over time and the corresponding ground truth state. The measured acceleration, which is not estimated over time but is instead copied to the state for each measurement, is not included.

$$\tilde{X}^{(i,t)} = \left[X^{(i,t)} \boxminus X_{\text{gt}}^{(i,t)} \right]_{1:9} \tag{3.16}$$

The normalized mean estimation error for dimension j is

$$\text{NMEE}_j^{(t)} = \frac{1}{50} \sum_{i=1}^{50} \frac{\tilde{X}_j^{(i,t)}}{\sqrt{\Sigma_{jj}^{(i,t)}}} \tag{3.17}$$

and the normalized estimation error squared

$$\text{NEES}^{(t)} = \frac{1}{50} \sum_{i=1}^{50} \tilde{X}^{(i,t)^T} \Sigma_{1:9,1:9}^{(i,t)^{-1}} \tilde{X}^{(i,t)}. \tag{3.18}$$

Because the position estimate is of particular interest, the NEES of the position component only is computed separately as well.

$$\text{NEES}_x^{(t)} = \frac{1}{50} \sum_{i=1}^{50} \tilde{X}_{4:6}^{(i,t)^T} \Sigma_{4:6,4:6}^{(i,t)^{-1}} \tilde{X}_{4:6}^{(i,t)}. \tag{3.19}$$

Results are plotted in Fig. 3.9 and 3.10. The NMEE plotted in Fig. 3.9 indicates that the estimator is unbiased. The 95%-probability intervals of the NEES averaged over 50 simulation runs are $[2.36, 3.72]$ for the 3-dimensional position estimate and $[7.85, 10.2]$ for the 9-dimensional $\tilde{X}^{(i,t)}$ [BKL02, chapter 1.5]. The NEES plotted in Fig. 3.10 are both below these intervals. Consequently, the estimator is slightly pessimistic, both overall and regarding the position in particular.

Since the covariance of the position prior is approximated, as in eq. (3.11), by spherical covariance of the form $\Sigma = \sigma^2 I$, the position covariance estimated by the Position-Prior-Filter is bounded by the approximated covariance of the position prior. This follows from Theorem 2 in Sect. 3.10.

Because empirically the Position-Prior-Filter provides conservative position error estimates, this leads to the conclusion that the position error itself is bounded. To further support this, I ran the filter on 2 hours of simulated data. The corresponding position estimation errors, plotted in Fig. 3.11, do not diverge, further supporting the boundedness of the position.

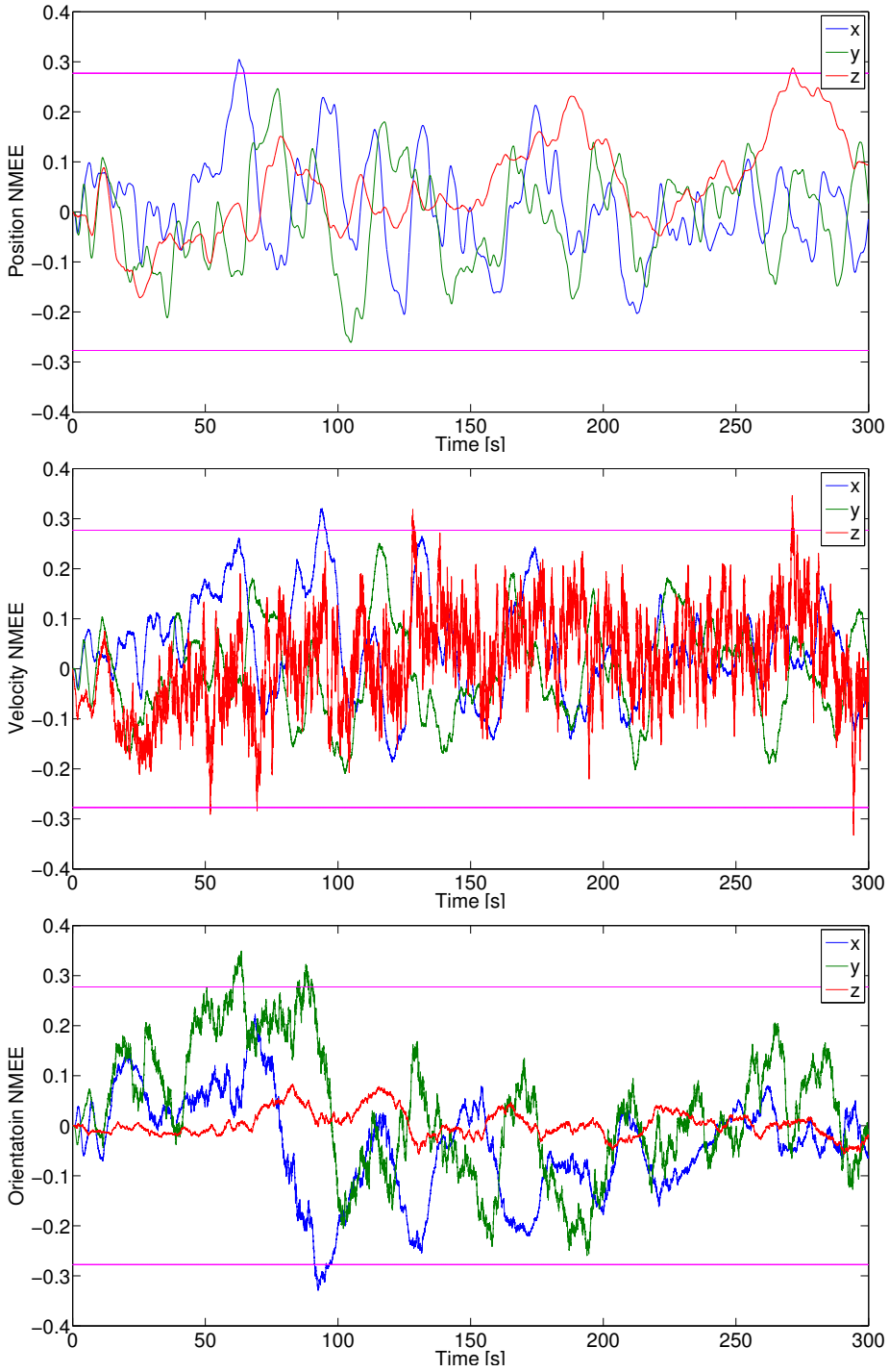


Figure 3.9 *Position-Prior-Estimator NMEE*
Normalized mean estimation error (NMEE) of the Position-Prior-Estimator of the position, velocity and orientation w.r.t. to the generated ground truth. The magenta line marks the 95% region. The estimates stay mostly in that region, indicating a consistent filter.

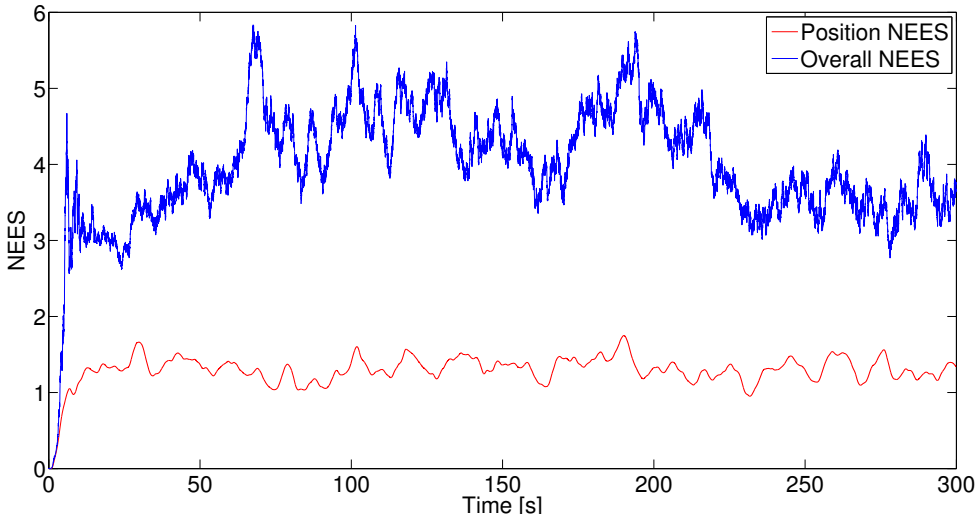


Figure 3.10 *Position-Prior-Estimator NEES*
Normalized Estimation Error Squared (NEES) both overall (orientation, position and velocity; in blue) and of the position component only. Both are below their 95%-interval ($[2.36, 3.72]$ for the 3-dimensional position, $[7.85, 10.2]$ for 9 overall dimensions), indicating a pessimistic estimator.

Due to periodicity, the orientation error is bounded automatically. As a by-product of the 2-hour experiment, I found that the orientation error, plotted in Figure 3.12 and again not considering the unobservable heading, also does not diverge. I ran the orientation estimation on the same data also using the Velocity-Prior- and the Acceleration-Prior-Filter. The data suggests that the orientation error increases a little, the more often the sensor data is integrated before applying a zero-prior. This effect is probably amplified by the simulation setup, which uses white noise acceleration, which makes the white noise approximation to the acceleration prior exact.

3.8 Discussion

In the previous sections I have argued that the classical orientation estimation from inertial sensor data really is estimation from inertial sensor data and prior information, namely that the average acceleration is zero. I presented two alternatives, the average velocity and position being zero, which led to two alternative reasonable orientation estimators. The latter were naturally extended to also estimate linear velocity and position, with the restriction that the prior information must refer to the desired quantity or its integral, but not its derivative.

With the Position-Prior-Filter, I presented a globally accurate orientation estimator (except for the heading), providing a locally, i. e. to some zero-position reference, accurate position estimate, which does not suffer from unbounded drift over longer estimation periods as integrating accelerometer measurements does.

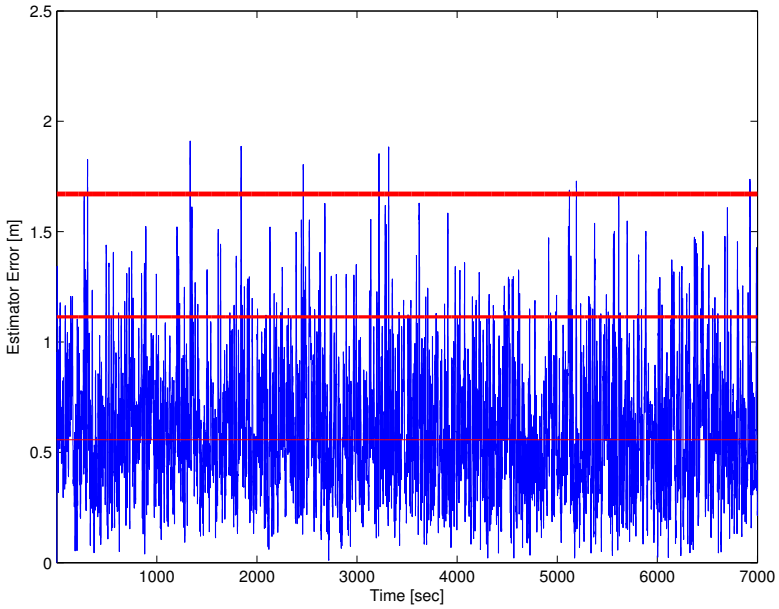


Figure 3.11 *Long Term Position Error*
Error of the estimated position (blue) over two hours of simulated data. The red lines mark the σ_ρ, $2\sigma_\rho$ and $3\sigma_\rho$ regions of the prior from eq. (3.10), displayed in Fig. 3.8.

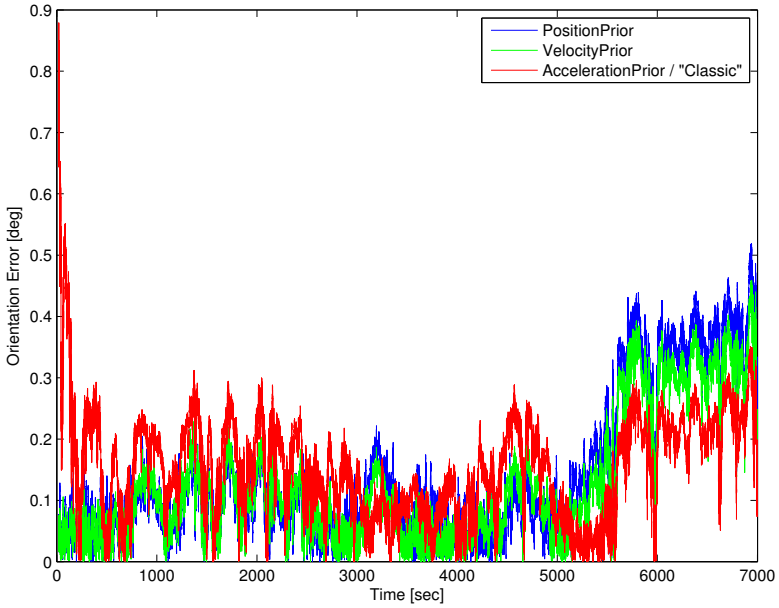


Figure 3.12 *Long Term Orientation Error*
Error of the estimated orientations (ignoring errors around the vertical axis) over two hours of simulated data. It appears that using a prior on integrated measurements increases the error, i. e. the velocity prior yields better orientation results than the position prior, but is worse than using the acceleration directly.

It is not entirely clear what the applications for this might be. Candidates are virtual-reality applications, where locally accurate positions of objects meeting the assumption about the position are of interest. With human posture estimation in mind, I tracked the pose of the IMU of my mobile phone while walking around the office floor. While the result drifts around zero, movements are clearly recognizable. Although there is no ground truth or other measurement to compare to, the rendering of the pose[4] serves as another illustration of the output of the Position-Prior-Filter.

A more immediate consequence is that if orientations are to be estimated, the inclination can be corrected without keeping the acceleration around if, for instance, the velocity is part of the state anyway. I made use of this when developing the posture estimator in Chap. 4.

3.9 Derivation of the Acceleration-Prior Result

To see why the zero-acceleration prior is equivalent to the measurement update of the original orientation estimator, I analytically derive the first-order approximations of both filters, i. e. using the scaled axis representation of the orientation instead of the rotation matrix and the $\boxplus/\boxminus$ operators.

The conventional filter's state is a 3D vector $r \in \mathbb{R}^3$, representing the orientation as a scaled axis. The first order approximation of the corresponding rotation matrix is $R \approx I + r_\times$, where $r_\times$ is again the matrix mapping any vector $v \in \mathbb{R}^3$ to the cross-product with r: $r_\times v = r \times v$.

The original orientation estimator estimates the parameters of the normal distribution $\mathcal{N}(r, \Sigma_{rr})$. When it receives an accelerometer measurement, $u_a \in \mathbb{R}^3$, it computes the conditional Gaussian distribution in the measurement update

$$r_{|z=u_a} = r + K(u_a - (1 - r_\times)(-g)) \tag{3.20}$$

$$\Sigma_{rr|z=u_a} = \Sigma_{rr} - K g_\times \Sigma_{rr} \tag{3.21}$$

with $K = \Sigma_{rr} g_\times^T \Sigma_Z^{-1}$, $\Sigma_Z = g_\times \Sigma_{rr} g_\times^T + \Sigma_{u_a}$ using the linearized measurement model $z = (1 - r_\times)(-g) + \delta$, with $\delta \sim \mathcal{N}(0, \Sigma_{u_a})$.

The zero-acceleration prior filter includes the acceleration in its state, i. e. it estimates the parameters of the Gaussian $\mathcal{N}(X, \Sigma)$ with

$$X = \begin{bmatrix} r \\ a \end{bmatrix} \quad \text{and} \quad \Sigma = \begin{bmatrix} \Sigma_{rr} & \Sigma_{ra} \\ \Sigma_{ar} & \Sigma_{aa} \end{bmatrix}.$$

When the zero-acceleration prior filter receives the same accelerometer measurement, u_a, it

[4] *http://www.informatik.uni-bremen.de/agebv2/downloads/videos/wenk_diss/rigidbodyposecoffee.mp4*

updates the state using the linear dynamic model

$$X' = \begin{bmatrix} r' \\ a' \end{bmatrix} = \begin{bmatrix} r \\ u_a + (1 - r_\times)g \end{bmatrix}. \tag{3.22}$$

I keep the acceleration in the sensor coordinate frame and rotate gravity in eq. (3.22). This makes the result very clear, despite working with a linearization. By expanding the acceleration, eq. (3.22) is split into state X and accelerometer measurement

$$X' = AX + Bu_a + \begin{bmatrix} 0 \\ g \end{bmatrix} \tag{3.23}$$

with

$$A = \begin{bmatrix} I & 0 \\ g_\times & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ I \end{bmatrix} .$$

According to eqs. (3.22) and (3.23), the acceleration is the last accelerometer measurement with the gravity-offset removed. The multiplication of $(1 - r_\times)g$ yields the first-order approximation of the gravity vector in the IMU's reference frame.

The accelerometer measurements are subject to zero-mean, Gaussian sensor noise with covariance Σ_{u_a} such that each measurement u_a is the actual acceleration set off by negative gravity plus the said noise. When applying the dynamic model to update the state as in eq. (3.23), the new covariance is calculated by propagating the state and measurement covariances through the dynamic model:

$$\begin{aligned} \Sigma' &= A\Sigma A^T + B\Sigma_u B^T \\ &= \begin{bmatrix} \Sigma_{rr} & \Sigma_{rr} g_\times^T \\ g_\times \Sigma_{rr} & g_\times \Sigma_{rr} {g_\times}^T + \Sigma_{u_a} \end{bmatrix}. \end{aligned} \tag{3.24}$$

Prior to all estimations I now assume an average acceleration $a_P = 0$ with some covariance, such that the acceleration experienced by the sensor is $a_P + \delta$ with $\delta \sim \mathcal{N}(0, \Sigma_P)$. The acceleration is related to the state by

$$\delta = a_P + \delta = \begin{bmatrix} 0 & -1 \end{bmatrix} X. \tag{3.25}$$

Note that since $a_P = 0$, the sign does not matter, but choosing a minus here eases the comparison of the end result to the original filter equations.

Using this relationship, the state is conditioned on the prior assumed acceleration. Since both the acceleration and its covariance are replaced by the next measurement in each dynamic update, the orientation component is considered only:

$$r_{|a_P=0} = r + K(u_a + (1 - r_\times)g) \tag{3.26}$$

$$\Sigma_{rr|a_P=0} = \Sigma_{rr} - K g_\times \Sigma_{rr} \tag{3.27}$$

with $K = \Sigma_{rr} g_\times^T [\Sigma_{a_P}]^{-1}$ and $\Sigma_{a_P} = g_\times^T \Sigma_{rr} g_\times + \Sigma_{u_a} + \Sigma_P$.

Eqs. (3.26) and (3.27) are the same as the measurement update, eqs. (3.20) and (3.21), of the original filter. The only difference is that Σ_{a_P}, which corresponds to the covariance of the expected measurement, has this extra Σ_P, which separates the covariance of actually measuring gravity from the sensor noise.

3.10 Upper Bounds on Noise Covariances

Theorem 1 (White Approximation to Autocorrelation)**.** *Let $X_1, \ldots, X_n$ be random variables in $\mathbb{R}^d$ with covariance norm $\| \operatorname{Cov}(X_i, X_j) \| \leq \sigma^2 \exp\left(-\frac{\delta t}{\tau}|i-j|\right)$, then for any series of samples $u^T = \begin{bmatrix} u_1^T \cdots u_n^T \end{bmatrix}$*

$$u^T \operatorname{Cov}(X_{1\ldots n})\, u \leq u^T u \sigma^2 \left(1 + 2\frac{\tau}{\delta t}\right), \tag{3.28}$$

where $\operatorname{Cov}(X_{1\ldots n})_{i,j} = \operatorname{Cov}(X_i, X_j)$

Proof.

$$\begin{aligned}
u^\top \operatorname{Cov}(X_{1\ldots n})\, u &= \sum_i \sum_j u_i^\top \operatorname{Cov}(X_i, X_j)\, u_j && (3.29)\\
&\leq \sum_i \sum_j |u_i^T \operatorname{Cov}(X_i, X_j)\, u_j| && (3.30)\\
&\leq \sum_i \sum_j \|u_i\|_2 \| \operatorname{Cov}(X_i, X_j) \|_2 \|u_j\|_2 && (3.31)\\
&= \bar{u}^T \overline{\operatorname{Cov}}(X_{1\ldots n})\, \bar{u} && (3.32)\\
&\leq \|\bar{u}\|_2 \|\overline{\operatorname{Cov}}(X_{1\ldots n})\|_2 \|\bar{u}\|_2 && (3.33)\\
&= \|u\|_2 \|\overline{\operatorname{Cov}}(X_{1\ldots n})\|_2 \|u\|_2 && (3.34)\\
&\leq u^T u \sigma^2 \left(1 + 2\frac{\tau}{\delta t}\right) && (3.35)
\end{aligned}$$

To get form eq. (3.30) to eq. (3.31) and from eq. (3.32) to eq. (3.33) the submultiplicativity of the 2-norm is used.

From eq. (3.29) and (3.31) it is clear that replacing the i'th and j'th vector-segments and (i,j)'th matrix-blocks by their norms provides an upper bound for the quadratic form $u^\top \operatorname{Cov}(X_{1\ldots n})\, u$. I denote the vector of segment-norms of u by $\bar{u}$ and the matrix of block-norms of $\operatorname{Cov}(X_{1\ldots n})$ by $\overline{\operatorname{Cov}}(X_{1\ldots n})$, s.t.

$$\bar{u} = [\|u_i\|]_{i=1}^n \text{ and } \overline{\operatorname{Cov}}(X_i, X_j) = [\| \operatorname{Cov}(X_i, X_j) \|]_{i,j=1}^n \quad . \tag{3.36}$$

The critical step from eq. (3.34) to eq. (3.35) is the approximation of $\|\overline{\operatorname{Cov}}(X_{1\ldots n})\|_2$ according to lemma 1. □

Lemma 1 (Upper bound of norm of exponentially decaying covariance)**.** *Let C be a matrix with $C_{i,j} \leq \sigma^2 \exp\left(-\frac{\delta t}{\tau}|i-j|\right)$, then $\|C\|_2 \leq \sigma^2\left(1+2\frac{\tau}{\delta t}\right)$.*

Proof. Since $\|C\|_2 \leq \sqrt{\|C\|_1\|C\|_\infty}$ [GV12, chapter 2] and C is symmetric, i. e. $\|C\|_1 = \|C\|_\infty$:

$$\begin{aligned}
\|C\|_2 &\leq \|C\|_\infty && (3.37)\\
&= \max_i \sum_j C_{i,j} && (3.38)\\
&\leq \max_i \sum_j \sigma^2 \exp\left(-\frac{\delta t}{\tau}|i-j|\right) && (3.39)\\
&\leq \sigma^2\left(1+2\sum_{k=1}^{\infty}\exp\left(-\frac{\delta t}{\tau}k\right)\right) && (3.40)\\
&= \sigma^2\left(1+\frac{2}{\exp\left(\frac{\delta t}{\tau}\right)-1}\right) && (3.41)\\
&\leq \sigma^2\left(1+\frac{2}{\frac{\delta t}{\tau}}\right) = \sigma^2\left(1+\frac{2\tau}{\delta t}\right) && (3.42)
\end{aligned}$$

The maximizing row of C, which gets us from eq. (3.39) to eq. (3.40), looks like a two-sided decaying pulse with its peak in the middle. i. e. it has $k = |i-j|$ in the middle, each element with $k > 0$ appears symmetrically both to the left and to the right of the peak, and it is bounded by two infinite exponentially decaying series.

The step from eq. (3.40) to eq. (3.41) is made by noticing that with $a_0 = q = \exp\left(-\frac{\delta t}{\tau}\right)$, $\sum_{k=1}^{\infty}\exp\left(-\frac{\delta t}{\tau}k\right)$ is a geometric series with $\lim_{n\to\infty} s_n = a_0 + a_0 q + \cdots = \frac{a_0}{1-q}$ [Bro+97, chapter 1]. Substituting a_0 and q yields eq. (3.41). □

Theorem 2 (Conditional Gaussian Covariance Bound)**.** *Let $x \sim \mathcal{N}(\mu, \Sigma)$ be a Gaussian random variable with covariance Σ, $z = h(x)$ the linear measurement model with $\frac{\partial h}{\partial x} = H = \text{const.}$, and let the covariance of the measurement be spherical $\Sigma_\rho = \sigma_\rho^2 I$. Then the covariance of that component $h(x)$ conditioned on the measurement, $H\Sigma_{|z_\rho}H^T$, is bounded by Σ_ρ: $\|H\Sigma_{|z_\rho}H^T\|_2 \leq \sigma_\rho^2$.*

Proof. Since $H\Sigma H^T$ is symmetric and positive-definite, it is decomposable into $H\Sigma H^T = R\Sigma_\Lambda R^{-1} = R\Sigma_\Lambda R^T$, with the diagonal eigenvalue matrix Σ_Λ and the corresponding orthonormal eigenvector matrix R.

The covariance of the conditional distribution [BKL02, chapter 1.4] is

$$\Sigma_{|z} = \Sigma - \Sigma H^T\left[H\Sigma H^T + \Sigma_\rho\right]^{-1} H\Sigma \tag{3.43}$$

which implies that

$$\begin{aligned} R^T H\Sigma_{|z}H^T R &= R^T H\Sigma H^T R && (3.44)\\ &\quad - R^T H\Sigma H^T \left[H\Sigma H^T + \Sigma_\rho\right]^{-1} H\Sigma H^T R && (3.45)\\ &= \Sigma_\Lambda - R^T H\Sigma H^T \left[H\Sigma H^T + \Sigma_\rho\right]^{-1} H\Sigma H^T R && (3.46)\\ &= \Sigma_\Lambda - \Sigma_\Lambda R^T \left[H\Sigma H^T + \Sigma_\rho\right]^{-1} R\Sigma_\Lambda && (3.47)\\ &= \Sigma_\Lambda - \Sigma_\Lambda \left[R^T H\Sigma H^T R + \Sigma_\rho\right]^{-1} \Sigma_\Lambda && (3.48)\\ &= \Sigma_\Lambda - \Sigma_\Lambda \left[\Sigma_\Lambda + \Sigma_\rho\right]^{-1} \Sigma_\Lambda = \Sigma_{\Lambda|z}. && (3.49) \end{aligned}$$

So,

$$H\Sigma_{|z}H^T = R\Sigma_{\Lambda|z}R^T, \tag{3.50}$$

whose 2-norm is its largest eigenvalue, which is one of the entries of the diagonal matrix $\Sigma_{\Lambda|z}$. Both $\Sigma_{\Lambda|z}$ and Σ_Λ may be of the same arbitrary size $n \times n$. Let $1 \leq \xi \leq n$. I denote the diagonal entries of Σ_Λ by σ_ξ^2.

According to eq. (3.49), each diagonal entry of $\Sigma_{\Lambda|z}$ has the form

$$\Sigma_{\Lambda|z,(\xi,\xi)} = \sigma_\xi^2 - \frac{\sigma_\xi^4}{\sigma_\xi^2 + \sigma_\rho^2} \leq \sigma_\rho^2, \tag{3.51}$$

so $H\Sigma_{|z}H^T$ has no eigenvalue larger than σ_ρ^2, i.e.

$$\|H\Sigma_{|z}H^T\|_2 \leq \sigma_\rho^2 = \|\Sigma_\rho\|_2. \tag{3.52}$$

□

Chapter 4

Posture Estimation Algorithm

Estimating the posture of a (human) skeleton is estimating the poses of all the individual bodies the skeleton is made of. However, in the applications of posture estimation, often also called motion capturing, the interest is usually more on the poses of the skeleton's bodies relative to each other. The poses, and the positions in particular, of the individual bodies relative to a globally fixed reference are secondary. Likewise, the skeleton's heading as a whole is usually not important. Given the magnetic field measurements taken on a shipyard, i.e. the targeted application environment, which were plotted in the very beginning in Fig. 1.2, there would not be much hope to recover the heading anyway.

The fact that the skeleton's bodies can not move completely independently, but are constrained by joints, provides most of the prior information to estimate the relative pose of two bodies, that are connected over a joint. The joint constraints are much stronger than the very weak assumption that an individual rigid body stays somewhere around a globally fixed reference point, which was the prior information in Chap. 3.

Judging from the available information, estimating a skeleton posture is easier than estimating the pose of a single rigid body. From an engineering perspective it is pretty hard. Since a single estimator has to estimate the orientations of all bodies at once, the dynamic and measurement models become more involved. Additionally, exploiting the skeleton's structure becomes computationally expensive as the number of bodies grows.

4.1 Posture from Motion

Posture from Motion [WF15] is my approach to recover the posture of a human wearing a suit equipped with one inertial sensor per body. The general idea is to measure the accelerations of jointed bodies and compensate for the accelerations caused by movements of the joint. These accelerations make one pair of vectors known in two coordinate systems. The second pair of vectors, which is necessary to compute the relative orientation of the two coordinate systems, is not obtained by using a complementary sensor. Due to the motion of the suit bearer, the measured accelerations change, most importantly in their direction. So *over time*, there is enough

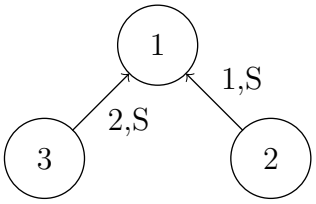


Figure 4.1 *Example Kinematic Tree.*
Kinematic tree with three bodies (vertices) and two joints (edges). A joint points from a body, the *successor body*, to the body's parent, the *predecessor body*.

information to determine the relative orientation from the acceleration measurements only, if there is a way to track the body and joint motion between two pairs of acceleration measurements. This is what the gyroscopes are used for.

Section 4.2 defines the functions to navigate the skeleton's structure, which is used to obtain the posture from orientations as described in Sect. 4.3. Sect. 4.4 contains the idea how to obtain complete relative orientations without magnetometers, which is implemented as an Extended Kalman Filter in Sect. 4.5 using the joint models of Sect. 4.6. How to decouple the estimation rate from the sensor sampling rate is the subject of Sect. 4.7. The resulting estimator's performance is evaluated in Sect. 4.8 and 4.9. The calibration of the skeleton is described in Sect. 4.10.

4.2 Skeleton Structure

To estimate the orientations of sensors on bodies subject to the structure of the skeleton the bodies are part of, a suitably formal representation of the skeleton is necessary. In robotics kinematic trees, as described by Featherstone [Fea08], are very popular and the formalism of choice here.

Bodies in the kinematic tree are represented by vertices, the edges between them represent joints. An simple example is drawn in Fig. 4.1. The tree gives rise to the following functions. With $\mathcal{B}$ being the set of bodies, $\mathcal{P}(\mathcal{B})$ the set of sets of bodies, and $\mathcal{J}$ being the set of joints, there are:

$$\lambda : \mathcal{J} \to \mathcal{B} \text{ with } \lambda(j) : \text{Predecessor of joint } j \in \mathcal{J} \tag{4.1}$$

$$\lambda_{\mathcal{B}} : \mathcal{B} \to \mathcal{B} \text{ with } \lambda_{\mathcal{B}}(b) : \text{Parent body of body } b \in \mathcal{B} \tag{4.2}$$

$$\mu : \mathcal{J} \to \mathcal{B} \text{ with } \mu(j) : \text{Successor of joint } j \in \mathcal{J} \tag{4.3}$$

$$\mu_{\mathcal{B}} : \mathcal{B} \to \mathcal{P}(\mathcal{B}) \text{ with } \mu_{\mathcal{B}}(b) : \text{Set of successor bodies of } b \in \mathcal{B} \tag{4.4}$$

Additionally, there is a one-to-one correspondence between bodies and inertial sensors.

For example, in the kinematic tree drawn in Fig. 4.1 we have $\lambda(2) = 1$, $\mu(2) = 3$, $\mu_{\mathcal{B}}(1) = \{2, 3\}$.

Additionally, every joint is labeled with a type. From the perspective of a posture estimator, the relative pose of two jointed bodies is somehow constrained. The set of constraints determines the

joint type. Theoretically, any combination of the DOF could be constrained, and the popular ones are listed in [Fea08, chapter 4]. I used only spherical joints, type S, and hinges, type H. Spherical joints, e. g. shoulders, fix the translation, hinges, e. g. elbows, also fix two of the three rotational degrees of freedom.

4.3 Obtaining Posture from Orientation Estimates

To estimate the posture of a skeleton, the relative poses of the skeleton's bodies that are connected via joints need to be estimated. To do so, it suffices to determine for each joint of the skeleton the orientation of the body succeeding the joint relative to the body preceding the joint (or vice versa).

If the positions $r_1, r_2 \in \mathbb{R}^3$ of the joint connecting two bodies B and $\lambda_{\mathcal{B}}(B)$ are known relative to both bodies' origins, then the pose of body B relative to body $\lambda_{\mathcal{B}}(B)$, represented by the homogenous coordinate transform $T_{\lambda_{\mathcal{B}}(B)\urcorner B} \in \mathbb{SE}(3)$, can be calculated by chaining the homogenous coordinate transform

$$T_{\lambda_{\mathcal{B}}(B)\urcorner B} = \begin{bmatrix} I & r_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} Q_{\lambda_{\mathcal{B}}(B)\urcorner B} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -r_2 \\ 0 & 1 \end{bmatrix}, \tag{4.5}$$

where $Q_{\lambda_{\mathcal{B}}(B)\urcorner B} \in \mathbb{SO}(3)$ is the orientation of body B relative to body $\lambda_{\mathcal{B}}(B)$.

This extends to all such pairs of bodies of the skeleton, so to get the posture, only the relative orientations need to be estimated. If one also wants to know the skeleton's orientation as a whole, it suffices to estimate the orientation of a single body of the skeleton in world coordinates.

4.4 Relative Orientation Estimation without Magnetometers

As mentioned in Chap. 2, the orientation of a rigid body is usually estimated using a Kalman Filter, which integrates measurements from an IMU and a magnetometer, both attached to the rigid body. The gyroscope measurements are integrated to follow short-term orientation changes and the accelerometer and magnetometer measurements to correct long term errors.

The measurement model of the accelerometer is based on the assumption that on long-term average negative gravity is measured. Because gravity always points down, this allows to determine the orientation except for the angle around the direction of gravity. This angle, the heading, is conventionally determined using the magnetometer.

To get rid of the magnetometer, known accelerations with directions different from the vertical, i. e. the direction of gravity, are needed. For this, there need to be accelerations, i. e. if the body is not moving at all, there's no way to get heading information out of the accelerometer. Even if the single rigid body accelerates, the acceleration is still unknown. But if a system of at least two

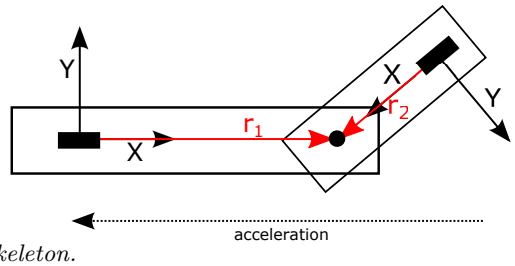


Figure 4.2 *Example Skeleton.*
Top-down view on two bodies ($B = 2, \lambda_{\mathcal{B}}(B) = 1$, rectangles) connected over a joint (circle), each equipped with an accelerometer (filled rectangles). The acceleration is measured by both sensors on different axes. This is the key idea how complete relative orientation can be recovered without magnetometers.

rigid bodies, that are both equipped with IMUs and are connected over a joint, accelerates, then, of course, both IMUs measure the same acceleration except for the motion along the degrees of freedom of their common joint.

Given that the system of rigid bodies does not rotate, the acceleration, which is exerted onto the system and measured by each IMU along different axes, determines their relative orientation, as in Fig. 4.2.

Note that there is still no heading information for the system of rigid bodies as a whole, i. e. it is not determined whether the system faces North or East. Their relative orientation, though, is now determined except for the angle around the direction of the acceleration, which may be different from the vertical. Given that the acceleration changes direction over time, the complete relative orientation becomes observable over time, because there is no permanently unobservable DOF left.

Joint Constraint

If the bodies do rotate, the situation is slightly more complicated due to the accelerations induced by the rotation. The accelerations induced by rotation are different at the two accelerometers, because they are separated by a displacement which acts as an additional lever arm. To account for this, one could calculate, for each rigid body of the skeleton, the accelerations virtual accelerometers positioned exactly at the joint locations would measure. For two bodies connected by a joint, the virtual accelerometers at the joint experience the same acceleration because they are at the same location, independent of the angular velocities the bodies may have.

Calculating the acceleration of a virtual accelerometer from a real accelerometer includes calculating the tangential acceleration $a_{\text{tangential}}$ over the displacement r from the real accelerometer to the joint, which is the cross-product $a_{\text{tangential}} = \dot{\omega} \times r$. The angular acceleration $\dot{\omega}$ is not measured directly. To avoid numerically differentiating the noisy gyroscope signals to obtain $\dot{\omega}$, I formulate the measurement model in terms of velocity. i. e. instead of differentiating the gyrometer measurement, the (real) accelerometer measurements are integrated.

For the bodies $\lambda_{\mathcal{B}}(B)$ and B, which are part of a skeleton, are connected over a common joint

and have relative orientation $Q_{\lambda_{\mathcal{B}}(B)\urcorner B} \in \mathbb{SO}(3)$ the following holds:

$$v^{(\lambda_{\mathcal{B}}(B))} + \omega^{(\lambda_{\mathcal{B}}(B))} \times r^{(\lambda_{\mathcal{B}}(B))} = Q_{\lambda_{\mathcal{B}}(B)\urcorner B} \left(v^{(B)} + \omega^{(B)} \times r^{(B)} \right), \tag{4.6}$$

where v are the velocities integrated from the accelerometer measurements, ω are the angular velocities measured by the gyrometers and r are the (constant and known) displacement vectors from the IMUs to the joint location.

4.5 Orientation Estimation Kalman Filter

State and Dynamics

To implement the above, I use an Extended Kalman Filter [BKL02, chapter 5] estimating the orientations of each body, where eq. (4.6) takes the role of the magnetometer measurement model. Since body velocities are needed for eq. (4.6), they join the bodies' orientations in the estimator state, such that the filter estimates the parameters of the Gaussian distribution $\mathcal{N}(X, \Sigma)$ with

$$X = \left[X^{(1)^T} \cdots X^{(N)^T} \right]^T \quad \text{with } X^{(k)} = \left[Q_{W\urcorner(k)}^T \;\; v^{(k)^T} \;\; b^{(k)^T} \right]^T \tag{4.7}$$

where $v^{(k)}$ is the k^{th} body's velocity in world coordinates, $b^{(k)}$ the bias of the gyroscope on body k, N is the number of bodies of the skeleton and $Q_{W\urcorner(k)} \in \mathbb{SO}(3)$ the orientation of body k in world coordinates. To estimate postures using workwear, the estimator is expected to be used for long periods. The gyroscopes' biases will probably change slightly during estimation, so the estimator keeps track of them.

Since there are again ordinary vectors mixed with orientations, I use $\boxplus$ analogously to eq. (3.4). For the state of body k the operators are, with $\delta x^{(k)^T} = \left[\delta q^T \;\; \delta v^T \;\; \delta b^T \right]$,

$$X^{(k)} \boxplus \delta x^{(k)} = \begin{bmatrix} Q_{W\urcorner(k)} \operatorname{Rot}(\delta q) \\ v^{(k)} + \delta v \\ b^{(k)} + \delta b \end{bmatrix} \quad X_2^{(k)} \boxminus X_1^{(k)} = \begin{bmatrix} \operatorname{aRot}\left(Q_{W\urcorner(k),(1)}^T Q_{W\urcorner(k),(2)} \right) \\ v_2^{(k)} - v_1^{(k)} \\ b_2^{(k)} - b_1^{(k)} \end{bmatrix} . \tag{4.8}$$

The complete state is just a stack of body states, so $\boxplus$ and $\boxminus$ for the complete state are repetitions of eq. (4.8), namely

$$X \boxplus \delta x = \left[X^{(k)} \boxplus \delta x^{(k)} \right]_{k=1}^{N} \quad X_2 \boxminus X_1 = \left[X_2^{(k)} \boxminus X_1^{(k)} \right]_{k=1}^{N} . \tag{4.9}$$

Two successive filter states X_i and X_{i-1} are related by the IMU measurements over the sampling time δt according to the dynamic model

$$\bar{X}_i = f(X_{i-1}, u_i) = \left[f'(X_{i-1}^{(k)}, u_i^{(k)}) \right]_{k=1}^{N} \tag{4.10}$$

with the dynamic input $u_i^{(k)} = \left[\omega_i^{(k)^T} a_i^{(k)^T}\right]^T$ consisting of the gyroscope and accelerometer measurements of the IMU attached to body k. f' updates the components of the state concerning body k

$$f'(X_{i-1}^{(k)}, u_i^{(k)}) = \begin{bmatrix} Q_{W^{\triangleleft}(k),(i-1)} \operatorname{Rot}\left(\left(\omega_i^{(k)} - b_{i-1}^{(k)}\right)\delta t\right) \\ v_{i-1}^{(k)} + \left(Q_{W^{\triangleleft}(k),(i)} a_i^{(k)} + g\right)\delta t \\ b_{i-1}^{(k)} \end{bmatrix}. \tag{4.11}$$

Here, g is again the gravitational acceleration. To update the covariance, the input's covariance is assembled from the noise densities, σ_ω and σ_a,

$$\Sigma_u = \begin{bmatrix} \Sigma_{u'} & & 0 \\ & \ddots & \\ 0 & & \Sigma_{u'} \end{bmatrix}, \quad \Sigma_{u'} = \begin{bmatrix} I_3\sigma_w^2/\delta t & 0 \\ 0 & I_3\sigma_a^2/\delta t \end{bmatrix}, \tag{4.12}$$

and propagated through the linearization of f with respect to the state, $A = \frac{\partial f}{\partial X_{i-1}}$, and the dynamic input, $B = \frac{\partial f}{\partial u_i}$:

$$\bar{\Sigma}_i = A\Sigma_{i-1}A^T + B\Sigma_u B^T + \Sigma_b \tag{4.13}$$

Σ_b is the tiny process noise which is added to the otherwise unmodified gyroscope biases,

$$\Sigma_b = \begin{bmatrix} \Sigma_{b'} & & 0 \\ & \ddots & \\ 0 & & \Sigma_{b'} \end{bmatrix}, \quad \Sigma_{b'} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & I_3\sigma_b^2\delta t \end{bmatrix} \in \mathbb{R}^{9\times 9}. \tag{4.14}$$

Because the state includes rotation matrices, the Jacobians A and B are a little trickier than usual. To arrive at a meaningful Jacobian, i. e. with the k'th column being the derivative with respect to the k'th dimension, I follow the familiar $\boxplus$-strategy to represent small changes vectorially. That is, A and B of eq. (4.13) are computed by

$$\frac{\partial f}{\partial X_{i-1}} \equiv A = \frac{\partial}{\partial \delta x}\left(f(X_{i-1} \boxplus \delta x, u_i) \boxminus f(X_{i-1}, u_i)\right) \tag{4.15}$$

$$\frac{\partial f}{\partial u_i} \equiv B = \frac{\partial}{\partial \delta u}\left(f(X_{i-1}, u_i + \delta u) \boxminus f(X_{i-1}, u_i)\right). \tag{4.16}$$

The actual Jacobians are given in Appendix A.

Measurements

Zero-Velocity Prior To compensate for accumulating drift, the measurement model of a classical orientation estimator expects the accelerometer to measure negative gravity plus noise, i. e. $-Q_{W^{\triangleleft}(k)}^T g + \delta$, $\delta \sim \mathcal{N}(0, \Sigma_\delta)$. Instead, as in Sect. 3.2, I augment the state and dynamic model with the velocity as in eq. (4.7) and eq. (4.11). Using $0 = v + \delta, \delta \sim \mathcal{N}(0, I_3\sigma_\delta^2)$ as a probabilistic prior yields the Velocity-Prior orientation estimator with $\sigma_\delta^2 = \sigma_{\delta,\rho}^2\left(1 + 2\frac{\tau}{\delta t}\right)$. As in Chap. 3, τ

is the decorrelation time and $\sigma^2_{\delta,\rho}$ the variance of two uncorrelated pseudo-measurements. δt is the time passed since the previous measurement update.

The relative orientations of adjacent bodies are determined using two joint model constraints, $h_j(X,\Omega)$ for all joints and $h_s(X)$ for hinges only. Both are explained in detail in Sect. 4.6. $\Omega = \left[w^{(k)}\right]_{k=1}^{N}$ contains the angular velocity measurements of the gyroscopes on the bodies. The value of these constraints is zero if they hold exactly, so the corresponding pseudo-measurements are

$$0 = h_j(X,\Omega) + \epsilon_j \text{ with } \epsilon_j \in \mathcal{N}(0,\Sigma_j)\,, \quad 0 = h_s(X) + \epsilon_s \text{ with } \epsilon_s \in \mathcal{N}(0,\Sigma_s) \quad . \tag{4.17}$$

Over time, this determines all relative orientations of the skeleton's bodies, but leaves the angle around the vertical of the skeleton as a whole undetermined, because no sensor provides information about the global heading. To prevent he corresponding covariance components from growing unboundedly, artificial information is added. I arbitrarily pick body 1 and assume that the y-component of the body's x-axis is 0 in world coordinates, $\left([0\ \ 1\ \ 0]Q_{W\curlywedge(1)}[1\ \ 0\ \ 0]^T\right) \sim \mathcal{N}\left(0,\sigma_z^2\right)$. This is almost equivalent to the angle around the vertical axis being 0. Choosing σ_z to be very large causes the estimate to slowly drift back to zero while still following short-term gyroscope measurements.

In summary, the prior model for the Kalman Filter's correction step is

$$0 = h(X,\Omega) + \zeta, \quad \zeta \sim \mathcal{N}(0,\Sigma_\zeta) \tag{4.18}$$

$$h(X,\Omega) = \left[v_{1\cdots N}^T \quad h_j(X,\Omega) \quad h_s(X) \quad [0\ \ 1\ \ 0]Q_{W\curlywedge(1)}[1\ \ 0\ \ 0]^T\right]^T \tag{4.19}$$

with measurements covariance

$$\Sigma_\zeta = \text{blkdiag}\left[\text{diag}\left(\overbrace{\sigma_\delta^2 \cdots \sigma_\delta^2}^{3N \text{ times}}\right), \Sigma_j, \Sigma_s, \sigma_z^2\right] \quad . \tag{4.20}$$

diag denotes the diagonal matrix with the entries on the diagonal as arguments, blkdiag is the analog for the block-diagonal matrix.

Using the linearizations of h,

$$H_X = \frac{\partial}{\partial \delta x} h(X \boxplus \delta x, \Omega) \quad H_\Omega = \frac{\partial}{\partial \Omega} h \tag{4.21}$$

the correction of the state $\bar{X}$ and covariance $\bar{\Sigma}$ is almost the standard linear Kalman Filter correction step:

$$K = \bar{\Sigma} H_X^T \left[H_X \bar{\Sigma} H_X^T + H_\Omega \sigma_\omega^2 \delta_t H_\Omega^T + \Sigma_\zeta\right]^{-1} \tag{4.22}$$

$$X = \bar{X} \boxplus K(0 - h(\bar{X})) \tag{4.23}$$

$$\Sigma = \bar{\Sigma} - K H_X \bar{\Sigma} \tag{4.24}$$

In (4.22), $H_\Omega \sigma_\omega^2 \delta_t H_\Omega^T$ enters the covariance, because h depends on the measured angular veloc-

ities, and based on linearization expresses how this increases the covariance of $h(X, \Omega)$. Also, (4.22) implies that the state, the orientations in particular, are not correlated with the current angular velocities. This obviously is an approximation, which becomes better, the more measurements are integrated between two correction steps.

4.6 Joint Models

Spherical Joints: $h_j(X, \Omega)$

To correct the relative orientations, eq. (4.6) is used for all body pairs connected over a joint as a probabilistic prior. Let there be M joints. For each joint $1 \leq j \leq M$, $\lambda(j)$ is the body preceding and $\mu(j)$ the body succeeding joint j, as defined in eqs. (4.1) and (4.3). Then the difference of the velocities determined by the IMUs on the bodies at joint j is

$$\left(v^{(\mu(j))} + Q_{W\urcorner\mu(j)}\psi^{(\mu(j))}\right) - \left(v^{(\lambda(j))} + Q_{W\urcorner\lambda(j)}\psi^{(\lambda(j))}\right) = J_j \tag{4.25}$$

with $\psi^{(l)} = (\omega^{(l)} - b^{(l)}) \times r_j^{(l)}$ for $l \in \{\lambda(j), \mu(j)\}$. Because the joint can not have two different velocities at the same time, eq. (4.6) holds. So for all $1 \leq j \leq M$

$$J_j + \epsilon = 0 \quad \text{with } \epsilon \sim \mathcal{N}(0, I_3\sigma_\epsilon^2), \quad \sigma_\epsilon^s = \sigma_{\epsilon,\rho}^2 \left(1 + 2\frac{\tau_\epsilon}{\delta t}\right), \tag{4.26}$$

where τ_ϵ is the decorrelation time constant for the joint constraint and $\sigma_{\epsilon,\rho}^2$ again the variance of two uncorrelated pseudo-measurements. Since (4.26) refers to a physical property and not to an assumption as to the system's motion, σ_ϵ^2 is much smaller than σ_δ^2 of the zero-velocity prior.

The pseudo-measurement sub-function $h_j(X, \Omega)$ is just a matter of stacking all joint velocity differences:

$$h_j(X, \Omega) = \left[J_1^T \dots J_M^T\right]^T \quad \Sigma_j = \operatorname{diag}(\overbrace{\sigma_\epsilon^2 \dots \sigma_\epsilon^2}^{3M \text{ times}}) \tag{4.27}$$

Revolute Joints / Hinges: $h_s(X)$

The relative motion of two bodies connected over a hinge is constrained to a single DOF, namely the rotation around the hinge axis.

In this orientation estimator, the orientations of jointed bodies, A and B, are defined by the orientations of the sensors mounted on them. When thinking about hinges, it is useful to think in physical coordinates, which are axis-aligned to the hinge axis. In such coordinates, A' for body A and B' for body B, the coordinate systems are related by a simple rotation of an angle α around the hinge axis a: $Q_{A'\urcorner B'} = \mathrm{Rot}(a\alpha)$. Thus, in physical coordinates, the hinge axis is an eigenvector of the relative orientation $Q_{A'\urcorner B'}$ of the two bodies, both in B' and A' coordinates: $a_{A'} = a = Q_{A'\urcorner B'}a = a_{B'}$.

Since the sensors are mounted approximately rigidly on the bodies, the orientation offsets between the physical and the usual sensor-defined coordinates, in this case $Q_{A\urcorner A'}$ and $Q_{B\urcorner B'}$, are constant. Using the orientation offsets, the hinge axis can be expressed in the sensor-defined coordinates:

$$a_A = Q_{A\urcorner A'} a \quad a_B = Q_{B\urcorner B'} a \tag{4.28}$$

which implies

$$a_A = Q_{A\urcorner A'} Q_{B\urcorner B'}^T a_B = Q_{A\urcorner B} a_B \ . \tag{4.29}$$

$Q_{A\urcorner B} = Q_{W\urcorner A}^T Q_{W\urcorner B}$ is easily computed from the orientation estimates in the state, so the hinge condition is

$$a_A = Q_{W\urcorner A}^T Q_{W\urcorner B}^T a_B \tag{4.30}$$

$$\Rightarrow 0 = Q_{W\urcorner A} a_A - Q_{W\urcorner B} a_B \ . \tag{4.31}$$

The statement that the hinge axis has the same coordinates in the world reference frame, no matter which sensor is used to calculate it, is no surprise, of course. It should be maintained approximately by the estimator.

According to eq. (4.31), the pseudo-measurement model for hinge joint j is

$$0 = h_{s,j}(X) + Q_{W\urcorner\mu(j)} \epsilon_{\mu(j)} + Q_{W\urcorner\lambda(j)} \epsilon_{\lambda(j)} \qquad h_{s,j}(X) = Q_{W\urcorner\mu(j)} a_j^{\mu(j)} - Q_{W\urcorner\lambda(j)} a_j^{\lambda(j)} \tag{4.32}$$

where $\epsilon_{\mu(j)} \sim \mathcal{N}\left(0, \Sigma_{\mu(j)}\right), \epsilon_{\lambda(j)} \sim \mathcal{N}\left(0, \Sigma_{\lambda(j)}\right)$ are Gaussian noise covering the uncertainty in the joint axis vectors. The total covariance for hinge joint j is

$$\Sigma_{s,j} = Q_{W\urcorner\mu(j)} \Sigma_\mu(j) Q_{W\urcorner\mu(j)}^T + Q_{W\urcorner\lambda(j)} \Sigma_\lambda(j) Q_{W\urcorner\lambda(j)}^T \ . \tag{4.33}$$

Stacking the contributions of the individual hinges yields the pseudo-measurement sub-function $h_s(X)$. Let L of the M joints be hinges, then

$$h_s(X) = \left[h_{s,1}^T \dots h_{s,L}^T\right]^T, \quad \Sigma_s = \text{blkdiag}(\Sigma_{s,1} \dots \Sigma_{s,L}) \ . \tag{4.34}$$

Deriving $\Sigma_{\mu(j)}$ and $\Sigma_{\lambda(j)}$ is a little involved, because the uncertainty of the joint axis is only in its direction, not in its length. $a_j^{\mu(j)}, a_j^{\lambda(j)} \in \mathbb{S}^2$ are directions in 3-dimensional space, or points on the unit sphere respectively, which have only 2 DOF.

Parameterization of $\mathbb{S}^2$. A 3-component unit vector is an over-parameterization of the hinge axis, similar to a rotation matrix being an over-parameterization of a 3-DOF orientation.

Points on a unit sphere, $\mathbb{S}^2$, are a $\boxplus$-manifold with the following operators given by Hertzberg et al. [Her+13].

$$\boxplus : \mathbb{S}^2 \times \mathbb{R}^2 \to \mathbb{S}^2, \quad x \boxplus \delta = R_x \exp \delta \tag{4.35}$$

$$\boxminus : \mathbb{S}^2 \times \mathbb{S}^2 \to \mathbb{R}^2, \quad y \boxminus x = \log\left(R_x^T y\right) \tag{4.36}$$

$$R_{\left[\begin{smallmatrix}x\\y\\z\end{smallmatrix}\right]} = \begin{bmatrix} x & -r & 0 \\ y & x\cos\alpha & -\sin\alpha \\ z & x\sin\alpha & \cos\alpha \end{bmatrix} \quad \exp\delta = \begin{bmatrix} \cos\|\delta\| \\ \operatorname{sinc}\|\delta\|\delta \end{bmatrix} \quad \log\begin{bmatrix} w \\ v \end{bmatrix} = \begin{cases} \operatorname{atan2}(0,w)e_1 & \text{if } v = 0 \\ \frac{\operatorname{atan2}(\|v\|,w)}{\|v\|}v & \text{otherw.} \end{cases} \tag{4.37}$$

where $\alpha = \operatorname{atan2}(z,y)$ and $r = \sqrt{y^2+z^2}$. R_x rotates the unit vector $e_1 = [1\ 0\ 0]^T$ to x. log is the inverse operation to exp. $\exp\delta$ is the vector one ends up at by moving away from e_1 on the surface of the sphere in the direction of δ until the new position on the sphere makes an angle of $\|\delta\|$ with e_1.

Hinge Covariance. The hinge axis is represented as a Gaussian random variable using a $\boxplus$-manifold element reference point and zero-mean, tangential noise as in eq. (2.28),

$$\mathcal{N}(a,\Sigma_a) \ \hat{=}\ a \boxplus \mathcal{N}\big(0,\bar{\Sigma}_a\big)\,, \tag{4.38}$$

with the mean axis $a \in \mathbb{S}^2$ and the 2-dimensional covariance matrix $\bar{\Sigma}_a \in \mathbb{R}^{2\times 2}$. I find the corresponding 3-dimensional covariance approximately by propagating the original covariance through the linearization of the $\boxplus$-operator at $\delta = 0$.

$$\Sigma_a = \operatorname{Cov}\left(a \boxplus \mathcal{N}\big(0,\bar{\Sigma}_a\big)\right) \approx \operatorname{Cov}(G_a\ \mathcal{N}\big(0,\bar{\Sigma}_a\big)) = G_a\bar{\Sigma}_a G_a^T \quad \text{with } G_a = \frac{\partial}{\partial\delta} a \boxplus \delta_{|\delta=0} \tag{4.39}$$

Because most of the derivative of $\boxplus$ vanishes at $\delta = 0$ [Her+13], $G_a \in \mathbb{R}^{(3\times 2)}$ is relatively simple to get. Expanding the above expression for G_a yields

$$G_a = \frac{\partial}{\partial\delta} R_a \begin{bmatrix} \cos\|\delta\| \\ \operatorname{sinc}\|\delta\|\delta \end{bmatrix}_{|\delta=0} = R_a \begin{bmatrix} \frac{\partial}{\partial\delta}\cos\|\delta\| \\ \frac{\partial}{\partial\delta}\operatorname{sinc}\|\delta\|\delta \end{bmatrix}_{|\delta=0} = R_a \begin{bmatrix} 0 \\ I \end{bmatrix}\ . \tag{4.40}$$

So G_a is just columns 2 and 3 of R_a as defined in eq. (4.37).

The 2-dimensional hinge covariance is spherical with variances $\sigma^2_{\text{hinge}} = \sigma^2_{\text{hinge},\rho}(1+2\frac{\tau_{\text{hinge}}}{\delta t})$, with the usual variance for uncorrelated pseudo-measurements $\sigma^2_{\text{hinge},\rho}$ and the decorrelation time constant τ_{hinge}. The resulting covariances for the hinge axes in sensor coordinates, as mentioned above for the model in eq. (4.32) are thus:

$$\Sigma_{\mu(j)} = G_{a_{j,\mu(j)}} G^T_{a_{j,\mu(j)}} \sigma^2_{\text{hinge}} \quad \text{and} \quad \Sigma_{\lambda(j)} = G_{a_{j,\lambda(j)}} G^T_{a_{j,\lambda(j)}} \sigma^2_{\text{hinge}} \tag{4.41}$$

4.7 Accumulating IMU Data: Decoupling Sampling and Estimation Rates

The Extended Kalman Filter, whose correction step (eqs. (4.22) to (4.24)) is computationally expensive due to the high dimensionality, is to be eventually run on an embedded device. For the application in the SIRKA suit, the required estimation rate is much lower than the sampling rate

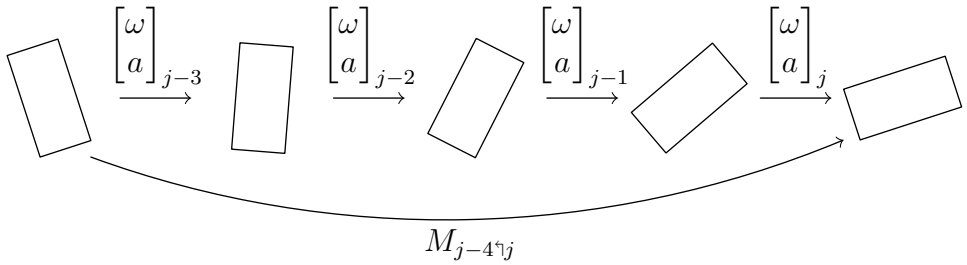


Figure 4.3 *Example Accumulation.*
The accumulate $M_{j-4\curvearrowright j}$ coalesces 4 raw IMU measurements. It maps directly from the state before the first measurement to the state after the most recent measurement. The intermediate states, which one would encounter if the raw measurements were integrated by the estimator one at a time, are "hidden" from the estimator.

of the sensors. So it is desirable to decouple those rates and only execute the high-dimensional operations with the lower frequency, i. e. update only every n^{th} sample.

Since dropping every other sensor measurement would result in loosing lots of information, I came up with the following technique to update only on every n^{th} sample, while still using every sensor measurement between two updates. The individual measurements are coalesced in measurement accumulates, such as in Fig. 4.3.

The angular velocity measurements are accumulated per IMU to a relative orientation and the acceleration measurements to a relative velocity at the sensor's sampling rate. The accumulation period is $\delta T = \delta t_1 + \cdots + \delta t_n$, because sampling is irregular (otherwise this would be $\delta T = n\delta t$). I modified to dynamic model accordingly to use such accumulates instead of the raw sensor data to update the state.

When an accumulate is computed, the orientation of the corresponding sensor is unknown. Thus, the accelerometer measurements can not be corrected for the negative gravity offset before they are integrated to the relative velocity. The latter looks as though the sensor accelerated upwards. However, when the accumulate is used in the estimator's dynamic update, the relative velocity is rotated into world coordinates, in which the negative gravity offset is known. Using the accumulation duration, the bogus upward velocity due to the negative gravity offset is easily computed and subtracted from the accumulated velocity. Accounting for the negative gravity offset later is possible, because it is a constant in the *global* coordinate system, which is constant as well. Compensating for the gyroscope bias, which is also approximately constant, will turn out to be much harder, because it is a constant in *local* sensor coordinates, which change as the sensor moves.

Accumulating IMU data is similar to accumulating a driving robot's odometry as a homogenous coordinate transform. Such a robot pose at step j is the matrix product of the pose from the previous step, $T_{0\curvearrowright j-1}$, and the odometry increment $T_{j\curvearrowright j-1}$: $T_{0\curvearrowright j} = T_{0\curvearrowright j-1} T_{j-1\curvearrowright j}$.

In analogy, I represent an IMU accumulate at step $j-1$ with the accumulated orientation Q and

velocity v by

$$M_{0\text{↰}j-1} = \begin{bmatrix} Q_{0\text{↰}j-1} & v_{0\text{↰}j-1} \\ 0 & 1 \end{bmatrix}. \tag{4.42}$$

An IMU increment is calculated from one sensor sample as

$$M_{j-1\text{↰}j} = \begin{bmatrix} \mathrm{Rot}\,(\omega\delta t) & \mathrm{Rot}(\omega\delta t)a\delta t \\ 0 & 1 \end{bmatrix}. \tag{4.43}$$

It is used to calculate the IMU accumulate at step j:

$$M_{0\text{↰}j} = M_{0\text{↰}j-1} M_{j-1\text{↰}j} \tag{4.44}$$

The relative accumulate, such as in Fig. 4.3, over an arbitrary number of samples of body k, n, can be recovered from two accumulates using the inverse of M.

$$\begin{aligned} M^{(k)}_{j-n\text{↰}j} = M^{(k)^{-1}}_{0\text{↰}j-n} M^{(k)}_{0\text{↰}j} &= \begin{bmatrix} Q^{(k)}_{j-n\text{↰}j} & v^{(k)}_{j-n\text{↰}j} \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} Q^{(k)^T}_{0\text{↰}j-n} Q^{(k)}_{0\text{↰}j} & Q^{(k)^T}_{0\text{↰}j-n} \left(v^{(k)}_{0\text{↰}j} - v^{(k)}_{0\text{↰}j-n} \right) \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{4.45}$$

When implementing this on a computer, there is a potential overflow in v, especially if the inclination of the body stays roughly the same for an extended period, because the accelerometer's $-g$ offset then accumulates in roughly the same direction. Hence, when computing a relative accumulate, it is crucial to compute the velocity difference in eq. (4.45) before rotating the vector. How the overflow is handled in the implementation of the accumulation is described in Sect. 5.4.

To use accumulates, the dynamic model from eq. (4.11) is adjusted as follows. The dynamic input concerning body k is now $u^{(k)}_i = M^{(k)}_{j-n\text{↰}j}$ and the update function of body k's components is

$$\bar{X}^{(k)}_i = f'(X^{(k)}_{i-1}, u^{(k)}_i) = \begin{bmatrix} Q_{W\text{↰}(k),(i-1)} Q^{(k)}_{j-n\text{↰}j} \,\mathrm{Rot}(-b^{(k)}_{i-1}\delta T) \\ v^{(k)}_{i-1} + Q_{W\text{↰}(k),(i-1)} \tilde{v}^{(k)}_{j-n\text{↰}j} + g\delta T \\ b^{(k)}_{i-1} \end{bmatrix}. \tag{4.46}$$

$$\text{with } \tilde{v}^{(k)}_{j-n\text{↰}j} = v^{(k)}_{j-n\text{↰}j} - \frac{\delta T}{2} b^{(k)}_{i-1} \times v^{(k)}_{j-n\text{↰}j} \tag{4.47}$$

This looks more complicated than it is. If there is no gyroscope bias, i. e. $b^{(k)}_{i-1} = 0$, then the update for a single body almost reduces to treating the body's state as an accumulate and "adding" the relative accumulate to it. The update for the velocity component accounts for the fact that the negative-gravity-offset of the accelerometer was also accumulated.

The function for the entire state is built from the function acting on the portion of the state pertaining to an individual body:

$$\bar{X}_i = f(X_{i-1}, u_i) = \left[f'(X^{(k)}_{i-1}, u^{(k)}_i) \right]^N_{k=1} \tag{4.48}$$

This looks exactly like the original definition, eq. (4.10), for raw sensor data, but uses eq. (4.46) and u_i is now the stack of accumulated sensor data, $u_i = \left[u_i^{(k)}\right]_{k=1}^{N}$.

With u now containing rotation matrices, the vectorial difference between two dynamic inputs has be to obtained and applied using $\boxminus$- and $\boxplus$-operators. This is necessary in particular to compute the Jacobian matrix of the dynamics function with respect to the dynamics input.

Accumulates are the Cartesian product of rotation matrices and 3-dimensional vectors (again, structurally equivalent to homogenous coordinate transforms) and thus are a $\boxplus$-manifold. Stacking the components rather than writing the matrix representation, the accumulate is just

$$m = \begin{bmatrix} Q \\ v \end{bmatrix} \quad \text{for } M = \begin{bmatrix} Q & v \\ 0 & 1 \end{bmatrix} \quad . \tag{4.49}$$

Using the $\boxplus/\boxminus$-operators for rotation matrices, the operators for accumulates are

$$m_2 \boxminus m_1 = \begin{bmatrix} Q_2 \boxminus Q_1 \\ v_2 - v_1 \end{bmatrix} \quad \text{and } m \boxplus \delta m = \begin{bmatrix} Q \boxplus \delta q \\ v + \delta v \end{bmatrix} \text{ with } \delta m = \begin{bmatrix} \delta q \\ \delta v \end{bmatrix} \quad . \tag{4.50}$$

Accordingly, the definition of the Jacobian matrix of the dynamics function with respect to the dynamics input, i. e. the accumulates, of eq. (4.16) changes, by substituting $+$ for $\boxplus$, to

$$\frac{\partial f}{\partial u_i} \equiv B = \frac{\partial}{\partial \delta u} \left(f(X_{i-1}, u_i \boxplus \delta u) \boxminus f(X_{i-1}, u_i) \right) \quad . \tag{4.51}$$

4.7.1 Correcting Accumulated Gyroscope Bias

If there is a gyroscope bias, the accumulate is approximately corrected in eq. (4.46) for the bias before it is added to the state. With $\tilde{\omega} = \omega - b$ being the unknown angular velocity measurement corrected for the bias, the incremental accumulate from eq. (4.43) would be

$$M_{j-1 \shortmid j} = \begin{bmatrix} \mathrm{Rot}\left((\tilde{\omega} + b)\delta t\right) & \mathrm{Rot}\left((\tilde{\omega} + b)\,\delta t\right) a \delta t \\ 0 & 1 \end{bmatrix}. \tag{4.52}$$

Since accumulation periods are short, i. e. less than a second long, the gyroscope bias is practically constant while incremental accumulates are chained to a bigger relative accumulate. By the time the bias correction is applied, the individual accumulate increments are lost. In order to correct for the bias, the relative accumulate is approximated by a single longer incremental accumulate:

$$M_{j-n \shortmid j} = M_{j-n \shortmid j-n+1} \ldots M_{j-1 \shortmid j} \approx \begin{bmatrix} \mathrm{Rot}\left((\bar{\omega} + b)\delta T\right) & v_{j-n \shortmid j} \\ 0 & 1 \end{bmatrix} \tag{4.53}$$

$\bar{\omega}$ is the average angular velocity corrected for the bias. I've already borrowed the matrix representation of accumulates from homogeneous coordinate transforms ($\mathbb{SE}(3)$). From $\mathbb{SE}(3)$ I borrow

also the matrix exponential map [II11], such that

$$M = \exp \begin{bmatrix} [\omega\delta T]_\times & u \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathrm{Rot}\,(\omega\delta T) & A(\omega\delta T)u \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Q & v \\ 0 & 1 \end{bmatrix} \ , \tag{4.54}$$

$$\text{where } A(q) = I + \frac{1-\cos\|q\|}{\|q\|^2}[q]_\times + \frac{\|q\|-\sin\|q\|}{\|q\|^3}[q]_\times^2 \ . \tag{4.55}$$

One strategy to correct for the bias would be to compute the rotation vector from the accumulate $q = \mathrm{aRot}(Q)$, compute $A(q)$, solve for u and use that to recompute a corrected accumulate.

The operations involved in that strategy are both rather expensive and lead to complicated derivatives. $A(q)$ in eq. (4.55) comes from the Taylor series of the matrix exponential

$$\exp \begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix} = I + \begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}^2 + \frac{1}{6}\begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}^3 \dots \tag{4.56}$$

$$\text{with } \begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}^2 = \begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}\begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} [q]_\times^2 & [q]_\times u \\ 0 & 0 \end{bmatrix} \tag{4.57}$$

$$\text{and } \begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}^3 = \begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}\begin{bmatrix} [q]_\times & u \\ 0 & 0 \end{bmatrix}^2 = \begin{bmatrix} [q]_\times^3 & [q]_\times^2 u \\ 0 & 0 \end{bmatrix} \ . \tag{4.58}$$

Reading off of eqs. (4.56) to (4.58), the series for $A\left((\bar{\omega}+b)\delta T\right)$ is

$$A\left((\bar{\omega}+b)\delta T\right) = I + \frac{1}{2}\left[(\bar{\omega}+b)\delta T\right]_\times + \frac{1}{6}\left[(\bar{\omega}+b)\delta T\right]_\times^2 + \dots \ . \tag{4.59}$$

Dropping all quadratic and higher-order terms, Au is

$$v = A\left((\bar{\omega}+b)\delta T\right)u \approx u + \frac{1}{2}\bar{\omega}\delta T \times u + \frac{1}{2}b\delta T \times u \ , \tag{4.60}$$

so the corrected velocity component, $\tilde{v}$, of the accumulate is approximately

$$\tilde{v} \approx v - \frac{1}{2}b\delta T \times u \ . \tag{4.61}$$

Computing u would require obtaining and inverting A. Approximating further by substituting u with $Au = v$ leads to

$$v - \frac{1}{2}b\delta T \times v = v - \frac{1}{2}b\delta T \times A\left((\bar{\omega}+b)\delta T\right)u \tag{4.62}$$

$$\approx v - \frac{1}{2}b\delta T \times u - \frac{1}{4}\delta T^2 b \times \bar{\omega} \times u - \frac{1}{4}\delta T^2 b \times b \times u \tag{4.63}$$

$$= \tilde{v} - \frac{1}{4}\delta T^2 b \times \bar{\omega} \times u - \frac{1}{4}\delta T^2 b \times b \times u \tag{4.64}$$

$$\approx \tilde{v} \ , \tag{4.65}$$

which is the same as the original approximation (4.61) except for cross- and quadratic terms, which are dropped when using the approximation in the update function in eq. (4.47).

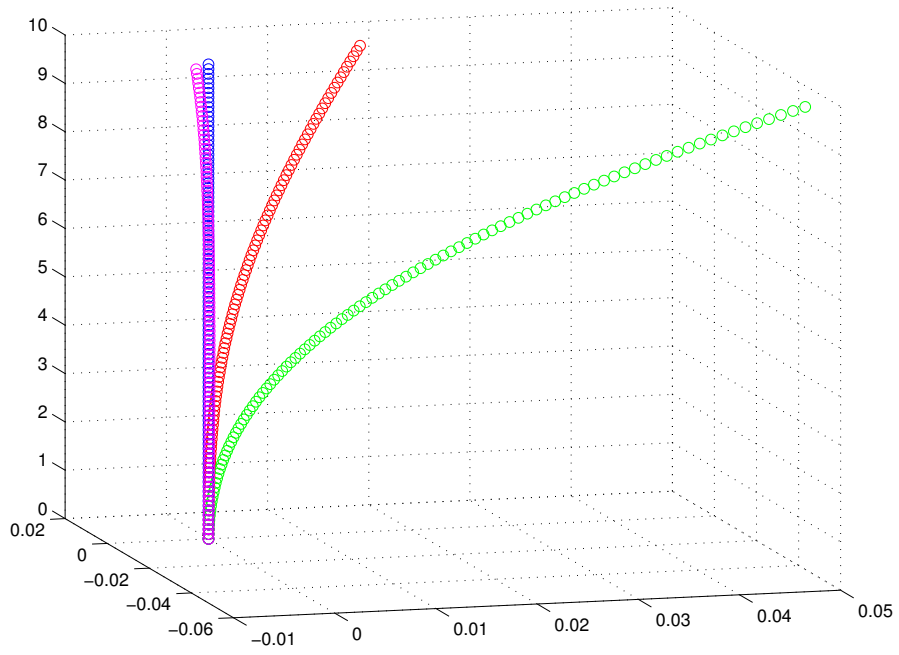


Figure 4.4 *Bias Correction of Accumulate Velocity*
Evolution of the velocity component of an accumulate over one second. The sensor does not accelerate, so the accelerometer measures negative gravity only, resulting in a straight velocity (blue). Adding the gyroscope bias causes a large error in the velocity (green). The velocity corrected (red) using the approximation implemented in the OFM estimator is worse than the velocity corrected using the more expensive procedure (magenta), but still substantially better than no correction.

The correction of the accumulated orientation, $\mathrm{Rot}\left((\bar{\omega}+b)\delta T\right)$, for the bias is more important, though simpler. Again dropping all quadratic and higher-order terms, the corrected orientation is obtained as follows.

$$\begin{aligned}
\mathrm{Rot}\left((\bar{\omega}+b)\delta T\right) &\approx I + [(\bar{\omega}+b)\delta T]_\times = I + [\bar{\omega}\delta T]_\times + [b\delta T]_\times && (4.66)\\
&= (I + [\bar{\omega}\delta T]_\times)(I + [b\delta T]_\times) - (\delta T)^2[\bar{\omega}]_\times[b]_\times && (4.67)\\
&\approx \mathrm{Rot}(\bar{\omega}\delta T)\,\mathrm{Rot}(b\delta T) && (4.68)\\
\Rightarrow \mathrm{Rot}(\bar{\omega}\delta T) &\approx \mathrm{Rot}\left((\bar{\omega}+b)\delta T\right)\mathrm{Rot}(-b\delta T) && (4.69)
\end{aligned}$$

Equation (4.69) is used in the dynamic update eq. (4.46). Figures 4.4 and 4.5 show how the approximate correction performs in comparison to the more expensive strategy mentioned in the beginning. Both figures are based on artificially biased, simulated data over one second. After this duration, differences are clearly visible. At the targeted estimation frequency of 10Hz, however, data is accumulated only over $(1/10)s$. Over this small duration, both methods are almost indistinguishable.

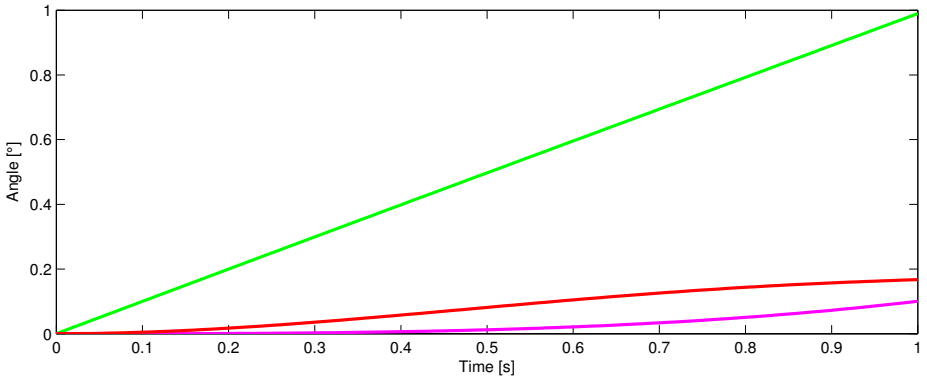


Figure 4.5 *Bias Correction of Accumulate Orientation*
Using the same data as in Fig. 4.4, the orientation error due to the bias (green) is reduced by both strategies, with the more expensive method (magenta) being only slightly better than the implemented, approximate method (red).

4.7.2 Accumulate Covariance

Because the the dynamic input $u_i^{(k)}$ changed, the input's covariance needs to change accordingly. It would be possible to accumulate the covariance of the accumulated orientation and velocity. Reconstructing the covariance of $u_i^{(k)}$, or equivalently $M_{j-n\urcorner j}$, from the covariances of $M_{0\urcorner j-n}$ and $M_{0\urcorner j}$ would involve multiplying a potentially overflowing quantity, corrupting the result. Instead, the covariance of $u_i^{(k)}$ is approximated directly using the accumulate itself. Given only the accumulate and not the individual measurements which led to it, the best guess is that the uncertainty of $u_i^{(k)}$ originates uniformly from each point in time over the accumulation period δT and that there were a constant angular velocity and acceleration.

The uncertainty in the accelerometer obviously leads to an uncertainty in the velocity component of $u_i^{(k)}$. Over the accumulation interval, this is

$$\Sigma_{u_i',a} = \begin{bmatrix} 0 & 0 \\ 0 & I_3 \sigma_a^2 \delta T \end{bmatrix} \quad . \tag{4.70}$$

The contribution due to the uncertainty of the gyroscope is a little trickier. At τ, a tiny error e due to gyro noise adds to the orientation. Additionally, it rotates the linear velocity accumulated after τ, $v(1-\frac{\tau}{\delta T})$, which is (almost) equivalent to adding $e \times v(1-\frac{\tau}{\delta T})$. Thus, at τ, the additional uncertainty is

$$\rho(\tau) = \sigma_\omega \begin{bmatrix} I_3 \\ [-v(1-\frac{\tau}{\delta T})]_\times \end{bmatrix} \quad . \tag{4.71}$$

Integrating $\rho(\tau)\rho(\tau)^T$ over the accumulation period yields the covariance contribution due to the

gyroscope uncertainty:

$$\Sigma_{u'_i,\omega} = \int_0^{\delta T} \rho(\tau)\rho(\tau)^T d\tau = \sigma_\omega^2 \begin{bmatrix} I_3\delta T & [v]_\times \frac{\delta T}{2} \\ [v]_\times^T \frac{\delta T}{2} & [v]_\times^T [v]_\times \frac{\delta T}{3} \end{bmatrix} \tag{4.72}$$

The covariance of $u_i^{(k)}$ is the sum of the two contributions:

$$\Sigma_{u'} = \Sigma_{u'_i,\omega} + \Sigma_{u'_i,a} = \delta T \begin{bmatrix} I_3\sigma_\omega^2 & \frac{1}{2}\sigma_\omega^2[v]_\times \\ \frac{1}{2}\sigma_\omega^2[v]_\times^T & \frac{1}{3}\sigma_\omega^2[v]_\times^T[v]_\times + I_3\sigma_a^2 \end{bmatrix} \tag{4.73}$$

The remainder of the covariance propagation through the decoupled dynamic model is analogous to (4.12) and (4.13), the correction step remains unchanged.

4.8 Evaluation: Accuracy and Influence of Accumulation

Before implementing the estimator for the actual SIRKA hardware, I tested the estimator using a preliminary implementation in MATLAB and early prototypes of the SIRKA sensor units. The prototypes shared the same bus; they could be queried synchronously with around 100Hz for raw, i. e. not accumulated, inertial sensor data.

To evaluate the estimator, I built a model skeleton of three rigid bodies connected by two ball-and-socket joints. For ground truth data, I used a commercial tracking system[1] to observe markers attached to the rigid bodies. On each body, I mounted one of the SIRKA prototype boards using double-faced tape. Fig. 4.6 pictures the setup. The displacement vectors to the joints were measured manually. The experiments with the model skeleton were rather short, which is why the gyroscope bias was not part of the estimator state. It has been calibrated in advance and subtracted from the raw measurements. Thus, in the estimator models, $b = 0$. To provide the motion the orientations are to be determined from, the skeleton arm was picked up from the ground and moved around for a few seconds.

Due to the SIRKA architecture, the sensors do not operate at a constant sampling frequency and are not electrically synchronized with the camera tracking system. They are synchronized among each other, though. Each prototype sensor board is equipped with a Bosch BMX055 IMU and a microcontroller sharing a data bus with the other sensor boards.

Sensor and ground truth data were synchronized a-posteriori by correlating angular velocity norms from the gyroscope and the camera tracking system.

Both without and with rate decoupling, i.e. working on raw and accumulated sensor data, two properties were tested. First, for a single body, the estimated orientation should be approximately the orientation observed by the tracking system, except for the unobservable angle around the vertical axis. Second, after the skeleton arm started to be moved, the estimates of the two relative

[1] ARTtrack/Dtrack2 from A.R.T. GmbH

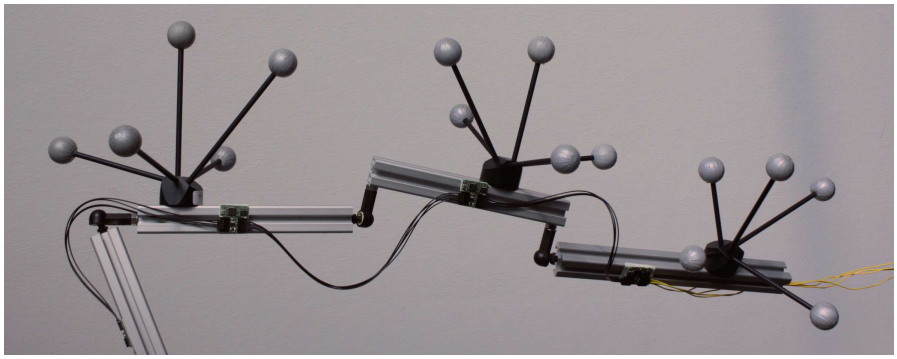


Figure 4.6 *Evaluation Arm*
The model skeleton used during the experiments. Spherical markers to be observed by the camera system are mounted on three bodies connected by ball-and-socket joints, each also carrying one SIRKA sensor board.

orientations between pairs of connected bodies should be approximately the relative orientations observed using the camera tracking system including the angle around the vertical.

To check the absolute orientation, I took both the estimate from IMU data, $Q^{(I)}_{W\curlywedge 2}$, and the ground truth orientation from the camera system, $Q^{(A)}_{W\curlywedge 2}$, and computed the inclination error angle, e, such that

$$e = \angle\left(Q^{(\mathrm{A})^T}_{W\curlywedge(2)}\,[0\ 0\ 1]^T\,,\ Q^{(\mathrm{I})^T}_{W\curlywedge(2)}\,[0\ 0\ 1]^T\right)\ . \tag{4.74}$$

This captures the orientation error except for the unobservable global heading. Including the latter in the error would add a completely arbitrary quantity and render the error useless.

The relative orientations were computed over the connecting joints, both from IMU and ground truth,

$$\begin{aligned} Q^{(I)}_{1\curlywedge 2} &= Q^{(I)^T}_{W\curlywedge 1} Q^{(I)}_{W\curlywedge 2}, \quad Q^{(I)}_{2\curlywedge 3} = Q^{(I)^T}_{W\curlywedge 2} Q^{(I)}_{W\curlywedge 3}, \\ Q^{(A)}_{1\curlywedge 2} &= Q^{(A)^T}_{W\curlywedge 1} Q^{(A)}_{W\curlywedge 2}, \quad Q^{(A)}_{2\curlywedge 3} = Q^{(A)^T}_{W\curlywedge 2} Q^{(A)}_{W\curlywedge 3}. \end{aligned} \tag{4.75}$$

4.8.1 Results without rate decoupling

I first calculated the quantities from (4.74) and (4.75) using the estimator without rate decoupling, obtaining estimates for each IMU measurement. This took the highly unoptimized MATLAB implementation of the estimator 633 seconds. Each estimate was associated with the ground truth datum closest in time according to the previously calculated synchronization.

The inclination error is plotted in Fig. 4.7. It is below 3 degrees, except for the initial error and the period immediately after the estimator corrected the relative orientations from the skeleton motion. The skeleton was at rest for the first 50 seconds. Since gravity always acts on the sensors, the estimate, with respect to the inclination, is expected to be slightly better if the sensors are at rest.

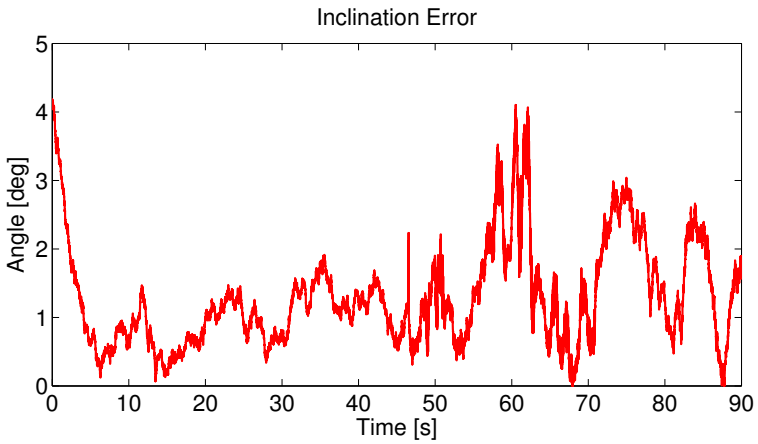


Figure 4.7 *Orientation error without rate decoupling.*
The graph shows the inclination error w.r.t. ground truth. It plots the angle between the world-vertical axis in sensor coordinates as obtained from the estimate and ground truth.

To determine the posture, the relative orientations of the bodies to each other are more interesting, and so are their errors. The errors about the first and second joint are the angular differences between the relative orientations obtained from the camera system and the IMUs as in eq. (4.75).

$$e_{\text{Joint1}} = \|q_{1\curvearrowright 2}\| \text{ s.t. } \text{Rot}(q_{1\curvearrowright 2}) = {Q_{1\curvearrowright 2}^{(A)}}^T Q_{1\curvearrowright 2}^{(I)} \tag{4.76}$$

$$e_{\text{Joint2}} = \|q_{2\curvearrowright 3}\| \text{ s.t. } \text{Rot}(q_{2\curvearrowright 3}) = {Q_{2\curvearrowright 3}^{(A)}}^T Q_{2\curvearrowright 3}^{(I)} \ . \tag{4.77}$$

The errors are plotted in Fig. 4.8. While the skeleton arm is at rest, the orientation errors stay approximately constant. The beginning of the movement is much more visible in the relative orientation errors, which include the rotations around the vertical, than it is in Fig. 4.7, because it is the moment the estimator can start to correct heading errors. In the first 10 seconds of movement, the orientation estimates for both joints are particularly bad. After about 10 seconds the estimator gets the orientation errors for both joints below 5 degrees. This rather long settle period may be caused by linearization with the large angular error of 20°.

4.8.2 Results with rate decoupling

To see how the rate decoupling technique affects the estimates, the same measurement series was fed to three accumulators that implement (4.43) and (4.44). At every $(n = 10)^{\text{th}}$ measurement, I used the current accumulate $M_{0\leftarrow 10k}$ to update the estimator with rate decoupling to obtain the k^{th} estimate. Thus, the estimation frequency was 10 times lower than the sampling frequency. This took an again highly unoptimized MATLAB implementation 73 seconds, i. e. it was 8.6 times faster.

Fig. 4.9 shows the the corresponding orientation error plot which should be approximately the orientation error obtained without rate decoupling. And indeed, Figs. 4.9 and 4.7 look almost identical.

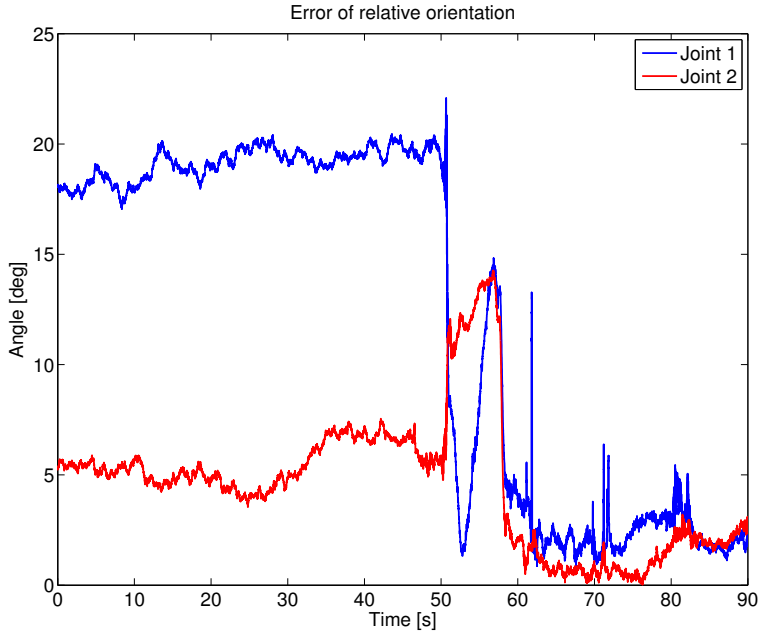


Figure 4.8 *Relative orientation errors.*
Errors of the relative orientations over joint 1 (blue) and joint 2 (red). After 50 seconds, the skeleton arm starts moving, after 60 seconds the orientation error drops considerably.

The same is true for the more interesting errors of the relative orientations over the two joints, plotted in Fig. 4.10, which again looks almost identical to the plot of the errors without rate decoupling, Fig. 4.8. So if the required estimation rate is only 1/10 of the sampling rate, there seems to be no obvious downside to using rate decoupling. To see the relative orientations themselves instead of the errors with respect to ground truth, watch the video[2] which accompanied the original publication [WF15].

[2] *http://www.informatik.uni-bremen.de/agebv/downloads/videos/wenk_iros_15.mp4*

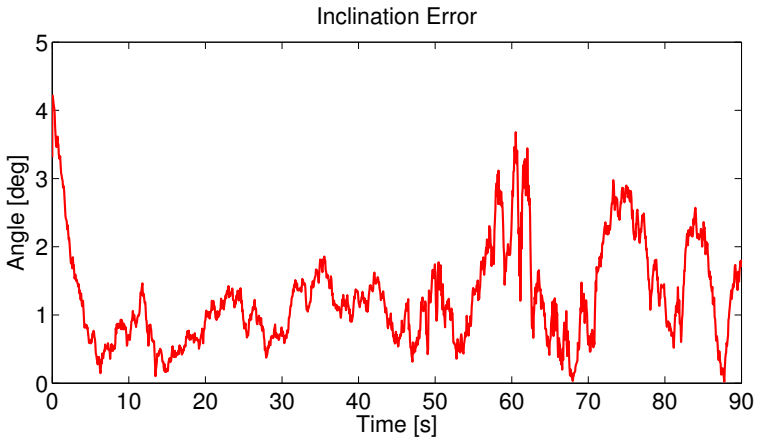


Figure 4.9 *Orientation error with rate decoupling.*
The graph shows the error of the inclination of an estimate obtained with rate decoupling enabled. The error is almost identical to the error of the estimator without rate decoupling.

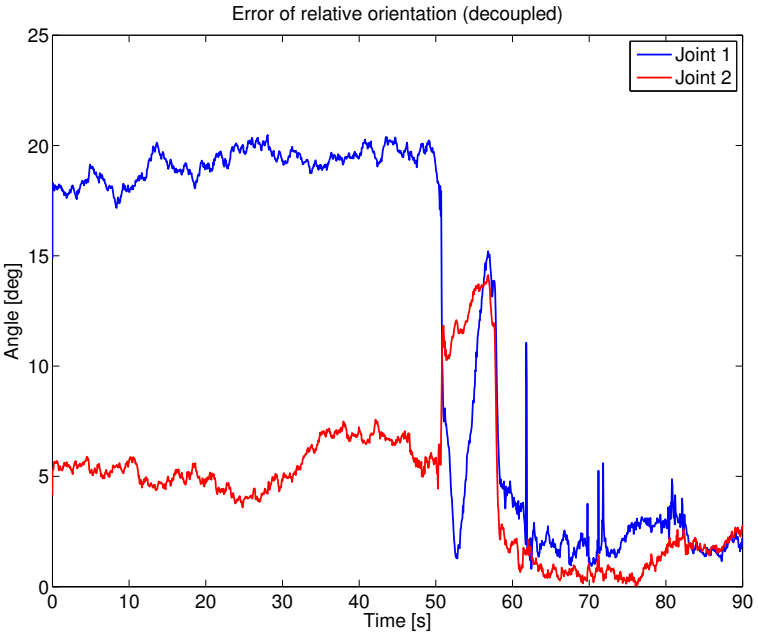


Figure 4.10 *Rate Decoupling Errors*
Errors of the relative orientations estimated with rate decoupling over joint 1 (blue) and joint 2 (red). The errors show the same characteristics as the errors without rate decoupling plotted in Fig. 4.8.

4.9 Evaluation: Estimating a Gyroscope Bias

In the implementation for the actual SIRKA suit, the gyroscope biases will be pre-calibrated. Before integrating measurements into accumulates, the measurements will be corrected using this calibration. In addition to the calibration, each gyroscope's bias is included in the estimator state, because the bias may change slightly while operating the suit.

To test how the estimator reacts to a gyroscope bias which is not considered when integrating measurements to accumulates, I generated accelerometer and gyroscope measurements from a simulated skeleton. The measurements of one sensor were corrupted by adding a bias of $2°/s$ on the gyroscope's x-axis.

The skeleton simulation moved each of the skeleton's DOF consecutively and subjected the whole skeleton to a sinusoidal motion along the world-x-axis, such that information to correct the relative orientations of adjacent bodies was provided. All gyroscope biases were initialized to zero when the estimator was initialized. As plotted in Fig. 4.11, the estimator was able to estimate the gyroscope bias from $(1/10)s$-accumulates.

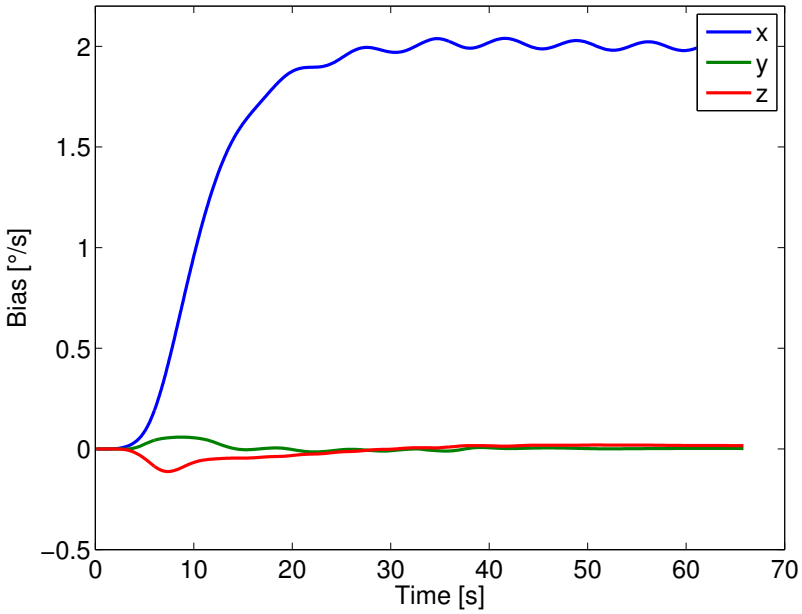
Due to the initially wrong bias, the estimator built up an error of approximately five degrees in the relative orientations to the preceding and succeeding body. However, as the estimate of the gyroscope's bias improved, the estimator recovered from the orientation error, which fell below half a degree after about $50s$.

4.10 Calibration of a Skeleton

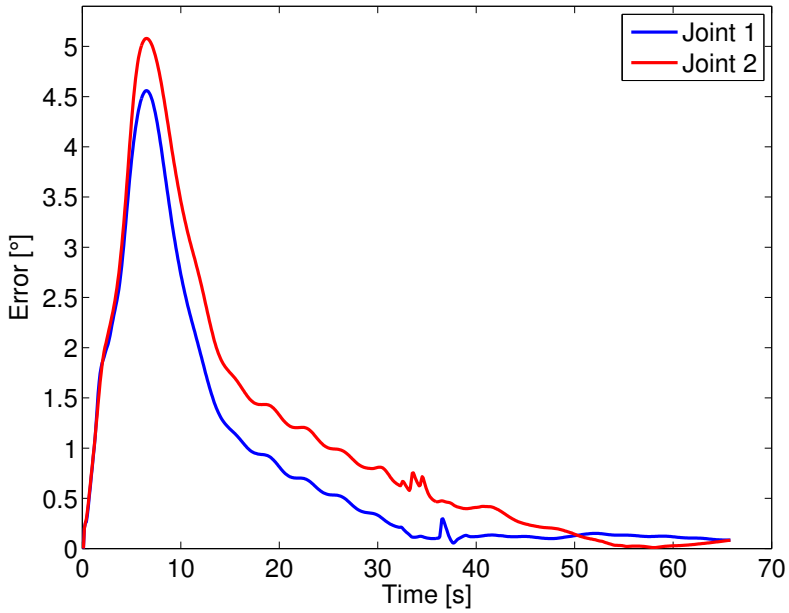
While it was possible to use a ruler to approximately measure the locations of the joints relative to the sensors on the metal arm displayed in Fig. 4.6, measuring manually is not an option for the intended application of the estimator, i. e. using sensors integrated into cloths. However, the joint locations and, for hinges, the joint axes can be calibrated automatically. Using the models I developed for the orientation estimator, finding the calibration can be stated as as least-squares problem, which can be solved by any least-squares solver that supports manifolds. For this work, I used SLOM [Her08]. That is, finding

$$\Theta = \arg\min_{\Theta} \frac{1}{2} \|F(Z, \Theta)\|_{\Sigma}^2 \tag{4.78}$$

where Θ is the parameters, Z the measurements and F an error function, which computes the difference with covariance Σ between the measurements of Z and the measurements predicted from the parameters Θ.



(a) Estimated Gyroscope Bias



(b) Orientation Error of a Biased Sensor

Figure 4.11 *Orientation Error and Gyroscope Bias*
Estimated gyroscope bias (top) and orientation errors (bottom) for a simulated sensor with artificially biased gyroscope measurements, $2°/s$ on the gyroscope x-axis. The errors of the relative orientations over the joints connecting the biased sensor to its adjacent sensors improve as the estimator acquires the gyroscope bias.

The parameters include the time-invariant calibration parameters and, as as by-product, the entire trajectories of all sensors. So Θ is the following for all joint locations $\mathcal{R}$, all hinge axes $\mathcal{A}$ and the trajectories $\mathcal{T}$.

$$\Theta = \begin{bmatrix} \mathcal{R} \\ \mathcal{A} \\ \mathcal{T} \end{bmatrix} \quad \text{with } \mathcal{R} = \begin{bmatrix} r_j^{\lambda(j)} \\ r_j^{\mu(j)} \end{bmatrix}_{j=1}^{M}, \quad \mathcal{A} = \begin{bmatrix} a_l^{\lambda(l)} \\ a_l^{\mu(l)} \end{bmatrix}_{l=1}^{L} \text{ and } \mathcal{T} = \begin{bmatrix} X_i \end{bmatrix}_{i=1}^{T} \tag{4.79}$$

There are again M joints, L of which are hinges. Every hinge axis $a \in \mathbb{S}^2$ is parameterized as a $\boxplus$-manifold as described in Sect. 4.6. The trajectories $\mathcal{T}$ of the sensors essentially are time series of the same states as in the Kalman Filter. However, since the movements for calibration are short, the gyroscope biases are approximately constant and thus left out of the parameter vector. They are accounted for when accumulating the sensor data.

Analogous to eq. (4.7), the sensor-related parameters of the least-squares problem at time step i are

$$X_i = \left[{X_i^{(1)}}^T \cdots {X_i^{(N)}}^T \right]^T \quad \text{with } X_i^{(k)} = \left[Q_{W^{\eta(k),i}}^T \; {v_i^{(k)}}^T \right]^T . \tag{4.80}$$

The measurements are the relative accumulates, which are also used in the dynamic model of the Kalman Filter, and the angular velocities at the end of each accumulation interval, which are used in the pseudo-measurement model of the orientation estimator. For each sensor k, $\mathcal{Z}$ has $T-1$ relative accumulates and angular velocities, that is

$$\mathcal{Z} = \begin{bmatrix} \mathcal{Z}_i \\ \Omega_i \end{bmatrix}_{i=2}^{T} \quad \text{with } \mathcal{Z}_i = \left[M_{i-1^{\eta i}}^{(k)} \right]_{k=1}^{N} \text{ and } \Omega_i = \left[\omega_i^{(k)} \right]_{k=1}^{N} . \tag{4.81}$$

Error Function

The heart of the calibration problem is the error function F. In addition to the difference between the predicted and actual measurements, F also considers the pseudo-measurements the orientation estimator uses to correct orientations:

$$F(\mathcal{Z}, \Theta) = \begin{bmatrix} F_{\text{vel}}(\Theta) \\ F_{\text{joint}}(\mathcal{Z}, \Theta) \\ F_{\text{hinge}}(\Theta) \\ F_{\text{rect}}(\Theta) \\ F_{\text{dyn}}(\mathcal{Z}, \Theta) \end{bmatrix} \tag{4.82}$$

F_{vel} contains the velocity prior for all sensors and time steps,

$$F_{\text{vel}}(\Theta) = \left[\left[v_i^{(k)} \right]_{k=1}^{N} \right]_{i=1}^{T} . \tag{4.83}$$

Joints F_{joint} represents all joint constraints and directly uses the joint constraint model from the orientation estimator, h_j from eq. (4.27), for all joints and all time steps $1 < i \leq T$ except the first,

$$F_{\text{joint}}(\mathcal{Z}, \Theta) = \Big[h_j(X_i, \Omega_i)\Big]_{i=2}^{i=T} \quad . \tag{4.84}$$

F_{hinge} is defined analogously to the hinge constraint function of the orientation estimator that uses the constraint functions in eq. (4.32). Instead of computing the difference in $\mathbb{R}^3$ as in eq. (4.32), I use the $\boxminus$-operation from eq. (4.36) of the parameterization of $\mathbb{S}^2$ to compute 2-dimensional difference between two joint axes.

Because the hinge axes are parts of the parameter vector, the calibrator has to be able to apply small changes to them, so it needs an implementation of of the $\boxplus/\boxminus$-operations anyway. This is not the case in the orientation estimator, which only uses but not estimates the hinge axes, and thus does not need any means to modify them. Thus, for the orientation estimator it was enough to lift the 2-dimensional uncertainty to a (3×3)-covariance matrix, without having to implement the $\boxplus/\boxminus$-operations on $\mathbb{S}^2$, which is not the case for the calibrator.

Analogous to eq. (4.32) the function for hinge $1 \leq j \leq L$ is

$$h_{s,j}(X, \mathcal{A}) = Q_{W \curvearrowright \mu(j)} a_j^{\mu(j)} \boxminus Q_{W \curvearrowright \lambda(j)} a_j^{\lambda(j)} \tag{4.85}$$

and for all L hinges and all time steps

$$F_{\text{hinge}} = \left[\Big[h_{s,j}(X_i, \mathcal{A})\Big]_{j=1}^{L}\right]_{i=1}^{T} \quad . \tag{4.86}$$

There is a rather technical problem with the joint locations of hinges, that I deal with using F_{rect}. Using inertial measurements only, the joint location of a perfect hinge can not be determined. During calibration, the information about the joint location r comes from the velocity, $v' = \omega \times r$, integrated at the sensor and caused by the angular velocity ω of the joint. In the coordinates of a sensor on a body attached to a hinge, ω changes only in length and never in direction. Thus, displacing r by $\delta r = \lambda\omega$, $\lambda \in \mathbb{R}$, in the direction of the hinge axis, and thus in the direction of the angular velocity ω, has no effect:

$$v' = \omega \times (r + \lambda\omega) = \omega \times r + \lambda\omega \times \omega = \omega \times r \tag{4.87}$$

However, since it has no effect in the calibration, it also does not matter for the orientation estimator. To remove the unobservable DOF, F_{rect} forces the joint location and the axis of each of the L hinges to be perpendicular:

$$F_{\text{rect}} = \begin{bmatrix} r_j^{\lambda(j)^T} a_j^{\lambda(j)} \\ r_j^{\mu(j)^T} a_j^{\mu(j)} \end{bmatrix}_{j=1}^{L} \tag{4.88}$$

Dynamics Measurements The orientation estimator uses the accumulated orientations and velocities to compute the prediction of the next estimator state. The calibrator uses the same

information to relate two consecutive states of a single sensor to each other. From two consecutive states of sensor (k), $X_{i-1}^{(k)}$ and $X_i^{(k)}$, the relative orientation and velocity are computed by

$$\bar{Q}_{i-1\text{↰}i}^{(k)} = Q_{W\text{↰}(k),i-1}^T Q_{W\text{↰}(k),i} \text{ and } \bar{v}_{i-1\text{↰}i}^{(k)} = Q_{W\text{↰}(k),i-1}^T \left(v_i^{(k)} - v_{i-1}^{(k)} \right) \ . \tag{4.89}$$

Both quantities are in the local coordinate system of sensor (k) at time step $i-1$. Both and the negative-gravity-offset make the relative accumulate between the two consecutive states. Thus, the error between two consecutive states and the corresponding relative accumulate over the duration $\delta T_i = t_i - t_{i-1}$,

$$M_{i-1\text{↰}i}^{(k)} = \begin{bmatrix} Q_{i-1\text{↰}i}^{(k)} & v_{i-1\text{↰}i}^{(k)} \\ 0 & 1 \end{bmatrix} \ , \tag{4.90}$$

is

$$h_{\text{dyn},i}^{(k)} \left(M_{i-1\text{↰}i}^{(k)}, X_i^{(k)}, X_{i-1}^{(k)} \right) = \begin{bmatrix} \text{aRot} \left({Q_{i-1\text{↰}i}^{(k)}}^T Q_{W\text{↰}(k),i-1}^T Q_{W\text{↰}(k),i} \right) \\ Q_{W\text{↰}(k),i-1}^T \left(v_i^{(k)} - v_{i-1}^{(k)} - g\delta T_i \right) - v_{i-1\text{↰}i}^{(k)} \end{bmatrix} \ . \tag{4.91}$$

This is almost the reverse of the dynamic model function of the orientation estimator in eq. (4.46). The differences are that, since the calibrator does not estimate gyroscope biases, there is no bias in eq. (4.91). Additionally, the orientation difference between the accumulated orientation and the orientation computed from the two states is a 3-dimensional vector.

From eq. (4.91), the dynamic measurement function of a single time step is assembled for all N sensors:

$$h_{\text{dyn}}(\mathcal{Z}_i, X_i, X_{i-1}) = \left[h_{\text{dyn},i}^{(k)} \left(M_{i-1\text{↰}i}^{(k)}, X_i^{(k)}, X_{i-1}^{(k)} \right) \right]_{k=1}^{N} \ . \tag{4.92}$$

The dynamic measurement function contributes to the error function F for all time steps except for the first, $1 < i \leq T$, because there obviously is no previous state for the first state to be related to. So,

$$F_{\text{dyn}} = \left[h_{\text{dyn}}(\mathcal{Z}_i, X_i, X_{i-1}) \right]_{i=2}^{T} \tag{4.93}$$

This completes the components of the error function F.

Covariance

The covariance Σ of $F(\mathcal{Z}, \Theta)$ is a block-diagonal matrix of the covariance of the individual components of F. The covariance is constructed similarly to the (pseudo-)measurement covariance in the correction step of the orientation estimator in eq. (4.20). That is,

$$\Sigma = \text{blkdiag}(\Sigma_{\text{vel}}, \Sigma_{\text{joint}}, \Sigma_{\text{hinge}}, \Sigma_{\text{rect}}, \Sigma_{\text{dyn}}) \ . \tag{4.94}$$

τ, τ_ϵ and τ_{hinge} are again the decorrelation time constants of the velocity prior, the joint constraint and the hinge constraint. Also as in the orientation estimator, $\sigma_{\delta,\rho}^2$, $\sigma_{\epsilon,\rho}^2$, $\sigma_{\text{hinge},\rho}^2$ are the variances uncorrelated pseudo-measurements would have. $\delta T_i = t_i - t_{i-1}$ is the time interval between two states in the trajectory.

As for the velocity prior in an orientation estimator measurement update, the covariance block

is

$$\Sigma_{\text{vel}} = \text{blkdiag}\left(\Sigma'_{\text{vel}}, \Sigma'_{\text{vel}}\left(1 + 2\frac{\tau}{\delta T_2}\right), \ldots, \Sigma'_{\text{vel}}\left(1 + 2\frac{\tau}{\delta T_T}\right)\right) \tag{4.95}$$
$$\text{with } \Sigma'_{\text{vel}} = \text{diag}(\overbrace{\sigma^2_{\delta,\rho} \cdots \sigma^2_{\delta,\rho}}^{3N \text{ times}}) \ .$$

The covariance block for the joints is analogously

$$\Sigma_{\text{joint}} = \text{blkdiag}\left(\Sigma'_{\text{joint}}\left(1 + 2\frac{\tau}{\delta T_2}\right), \ldots, \Sigma'_{\text{joint}}\left(1 + 2\frac{\tau}{\delta T_T}\right)\right) \tag{4.96}$$
$$\text{with } \Sigma'_{\text{joint}} = \text{diag}(\overbrace{\sigma^2_{\epsilon,\rho} \cdots \sigma^2_{\epsilon,\rho}}^{3M \text{ times}}) \ .$$

The hinge covariances are a lot simpler for the calibrator than they are for the orientation estimator, because they do not have to be lifted to (3×3)-blocks.

$$\Sigma_{\text{hinge}} = \text{blkdiag}\left(\Sigma'_{\text{hinge}}, \Sigma'_{\text{hinge}}\left(1 + 2\frac{\tau}{\delta T_2}\right), \ldots, \Sigma'_{\text{hinge}}\left(1 + 2\frac{\tau}{\delta T_T}\right)\right) \tag{4.97}$$
$$\text{with } \Sigma'_{\text{hinge}} = \text{diag}(\overbrace{\sigma^2_{\text{hinge},\rho} \cdots \sigma^2_{\text{hinge},\rho}}^{2L \text{ times}}) \ .$$

The variance of each perpendicularity constraint of a hinge's axis and location is just 1, since the constraint is entirely technical. Since there are L hinges, the covariance of the perpendicularity constraint is the $(2L \times 2L)$ identity matrix.

$$\Sigma_{\text{rect}} = I \ . \tag{4.98}$$

The covariance for a single relative accumulate is computed exactly as in the orientation estimator in eq. (4.73). For time step i and body k that is

$$\Sigma_{\text{dyn},i,k} = \begin{bmatrix} I\sigma^2_\omega & \frac{1}{2}\sigma^2_\omega [v^{(k)}_{i-1 \urcorner i}]_\times \\ \frac{1}{2}\sigma^2_\omega [v^{(k)}_{i-1 \urcorner i}]^T_\times & \frac{1}{3}\sigma^2_\omega [v^{(k)}_{i-1 \urcorner i}]^T_\times [v^{(k)}_{i-1 \urcorner i}]_\times + I\sigma^2_a \end{bmatrix} \delta T_i \ . \tag{4.99}$$

$\Sigma_{\text{dyn},i,k}$ is a block of the covariance matrix $\Sigma_{\text{dyn,i}}$ for time step i and all bodies $1 \leq k \leq N$,

$$\Sigma_{\text{dyn,i}} = \text{blkdiag}(\Sigma_{\text{dyn},i,1}, \ldots, \Sigma_{\text{dyn},i,N}) \ , \tag{4.100}$$

which in turn is a block of the covariance of all dynamic measurements:

$$\Sigma_{\text{dyn}} = \text{blkdiag}(\Sigma_{\text{dyn},2}, \ldots, \Sigma_{\text{dyn},T}) \tag{4.101}$$

As in the orientation estimator, the covariances of the velocity prior and the constraints become smaller as δT increases, because the corresponding quantities are assumed to be less correlated if further apart. In contrast, the dynamic measurement covariance increases with δT, as the uncertainty due to sensor noise adds up.

Initial Guess

Sensor Trajectory To avoid making the calibration process unnecessarily hard, the initial guess of the parameters Θ should be compatible with what is known at the outset. Not moving the sensors in the first $(1/10)s$ makes them integrate negative gravity in the first accumulate. The velocity component of the first accumulate is thus approximately $v = [0 \quad 0 \quad 1]^T (m/s)$.

For each sensor (k), the first accumulate is

$$M_{1\text{↰}2}^{(k)} = \begin{bmatrix} Q_{1\text{↰}2}^{(k)} & v_{1\text{↰}2}^{(k)} \\ 0 & 1 \end{bmatrix} . \tag{4.102}$$

Using the solution, eq. (2.22), to Wahba's problem in eq. (2.21), the initial guess of the first orientation of sensor (k) is

$$Q_{W\text{↰}(k),1} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det U \det V \end{bmatrix} V \text{ with } USV = \text{SVD}\left(\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T v_{1\text{↰}2}^{(k)^T} \right) , \tag{4.103}$$

where SVD denotes the singular value decomposition. From eq. (4.103), no information about the initial heading of sensor (k) is obtained, but the inclination in $Q_{W\text{↰}(k),1}$ is approximately correct. The initial velocity of sensor (k) is $v_i^{(k)} = 0$.

Starting from there, the sensor's trajectory approximately satisfies the dynamics. Thus for all time steps $1 < i \leq T$, the initial guesses for the orientations and velocities of sensor (k) are obtained recursively:

$$Q_{W\text{↰}(k),i} = Q_{W\text{↰}(k),i-1} Q_{i-1\text{↰}i}^{(k)} \quad \text{and } v_i^{(k)} = v_{i-1}^{(k)} + Q_{W\text{↰}(k),i-1} v_{i-1\text{↰}i}^{(k)} + g\delta T_i \tag{4.104}$$

$Q_{i-1\text{↰}i}^{(k)}$ and $v_{i-1\text{↰}i}^{(k)}$ are taken from the accumulates $M_{i-1\text{↰}i}^{(k)}$ and δT_i is the time interval over which the data has been accumulated. This is equivalent to the dynamic model of the Kalman Filter in eq. (4.46) without the bias correction. Using eq. (4.104), the trajectories of all bodies $1 \leq k \leq N$ are initialized approximately correct except for the rotation around the direction of the gravity vector g.

Hinges The initial guess of the hinge axes is obtained using a method suggested in a paper by Seel et al. [SSR12], which I already mentioned in Sect. 2.2. They noticed that the projections of the angular velocities of two adjacent sensors into the plane perpendicular to the hinge axes must be the same. So for a hinge joint j, Seel et al. came up with the following constraint:

$$\|\omega_i^{\lambda(j)} \times a_j^{\lambda(j)}\| - \|\omega_i^{\mu(j)} \times a_j^{\mu(j)}\| = 0 \tag{4.105}$$

In contrast to the hinge constraint in eq. (4.85), the signs of the axes are undetermined in eq. (4.105). However, eq. (4.105) does not depend on the relative orientation of the bodies, leading to the following very simple least-squares problem for hinge joint j over all time steps

$1 \leq i \leq T$:

$$\begin{bmatrix} a_j^{\lambda(j)} & a_j^{\mu(j)} \end{bmatrix} = \arg \min_{a_j^{\lambda(j)}, a_j^{\mu(j)} \in \mathbb{S}^2} \frac{1}{2} \sum_{i=1}^{T} \left(\|\omega_i^{\lambda(j)} \times a_j^{\lambda(j)}\| - \|\omega_i^{\mu(j)} \times a_j^{\mu(j)}\| \right)^2 \tag{4.106}$$

Before the actual calibration, I solve this small least-squares problem for each hinge joint j, also using SLOM, to provide an initial guess for the hinge axes. If $a_j^{\lambda(j)}$ and $a_j^{\mu(j)}$ came out with different signs, it would stand out in the main calibration problem. However, I avoid this problem in practice as suggested in [SSR12] by mounting the sensors such the z-axes of bodies connected over a hinge joint in approximately the same direction.

Chapter 5

Posture Estimation Application: Sensor Suit

The orientation estimator of the previous Chap. 4 is the heart of SIRKA, a sensor suit to provide the suit wearer with activity feedback. The sensors and all the other electronics are integrated into the suit, making SIRKA a self-contained system. To estimate the suit wearer's posture, the previously developed algorithm needs a suitable approximation of the person's skeleton, which is the subject of Sect. 5.1. Following that, I describe both the hard- and software design in Sect. 5.2 and 5.3–5.6, respectively.

In addition to simulations to quantitatively test the algorithm in the suit-configuration, I qualitatively evaluated the suit in its target setting, i. e. estimating postures of a person working on a shipyard, e. g. as in Fig. 1.1. The results are presented in Sect. 5.7–5.11.

5.1 Skeleton Structure

The human skeleton is approximated by a collection of jointed rigid bodies. The suit does not cover feet, hands and head, so these bodies are not included in the SIRKA skeleton. The knees are approximated by hinges, which connect tibiae and thighs. Elbows are also approximated by hinges, connecting the lower and upper arms. All other joints are represented as spherical joints. The upper arms are connected to approximately rigid bodies between neck and shoulders. Legs are connected to the lower end of the of the approximate spine, whose closest match in a real, natural human skeleton would be the Sacrum.

A human spine is a chain of tiny bones called vertebrae, which are connected using spherical joints. However, those spherical joints can not be moved completely independently. Given the relative orientations of the the vertebrae of a relaxed spine, the deviations from those relaxed orientations are usually highly correlated. Additionally there is little hope for the sensors integrated in the jacket, which dangles over the back of the wearer of the suit, to be properly aligned with the spine. Consequently, the spine is approximated by a chain of only 5 rigid bodies with no restriction in their relative orientation.

In total, there are fifteen bodies and fourteen joints, four of which are hinges.

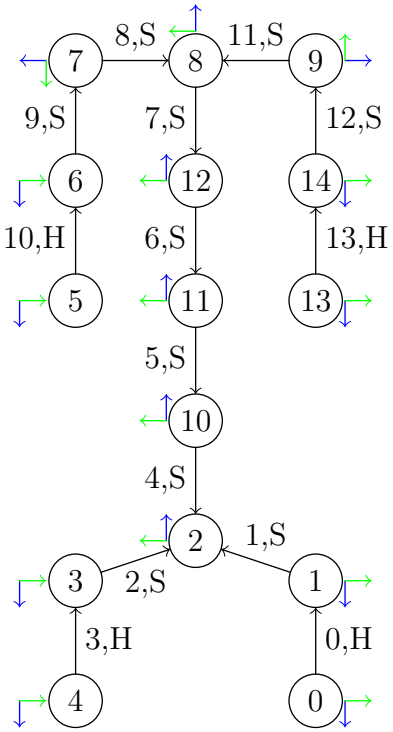


Figure 5.1 *SIRKA suit kinematic tree*
As in Fig. 4.1, vertices denote bodies and edges the joints connecting them, pointing from the *successor* to the *predecessor*. Additionally, each body is annotated with its physical coordinate system in a perfectly upright posture with arms hanging downwards. These coordinate systems are not important for the orientation estimator. They are used when the a posture estimate is sent or written. The z-axes are drawn blue, y-axes green. The x-axes make all the coordinate systems right handed, i. e. point into the paper.

The complete kinematic tree built as in Sect. 4.2 of the SIRKA suit is drawn in Fig. 5.1. Readers of [Fea08] might notice the unorthodox numbering. The conventional scheme is to label a node with a number smaller than the smallest label in any if its subtrees. Such a property is neither used in the orientation estimator nor in the calibrator. The numbering for SIRKA is inherited from the first, simple prototype mentioned earlier in Chap. 1, mainly to maintain compatibility with software developed by other partners of the SIRKA consortium. In the first prototype, the labels of the bodies corresponded to the hardware addresses of the sensors mounted on them.

Joint Sensor Map

The kinematic tree is stored in both human- and machine-readable form in a data structure called *Joint-Sensor Map (JSM)*. The JSM has two entries per joint which represent its location and, in case the joint is a hinge, axis in predecessor- and successor-coordinates, respectively. It is used as input both for the calibrator and for the orientation estimator. The calibrator ignores

Figure 5.2 *Inertial Motion Capturing Jacket*
This is the jacket of the motion capturing workwear SIRKA. It is equipped with a small embedded PC and inertial sensors. The inertial sensors are mounted inside the hook-and-loop-tape packages (tiny black squares) on the inside of the suit. The sensor data is fed to the Posture-from-Motion estimator to obtain a posture estimate of the suit's wearer.

the actual values for axes and locations and puts out a modified JSM, where the joint locations and hinge axes are replaced with the calibration result. Syntactically, the entries for hinge joint j in the JSM look like this:

$$\begin{array}{cccccc} j & \lambda(j) & r_j^{\lambda(j)^T} & \texttt{'H'} & a_j^{\lambda(j)^T} \\ j & \mu(j) & r_j^{\mu(j)^T} & \texttt{'H'} & a_j^{\mu(j)^T} \end{array}$$

5.2 Hardware

The SIRKA suit is made of the following components. There is, of course, the clothing itself, which is rofa's model 1002097, mostly as used by the Meyer Werft, with some extensions made by rofa to contain wires and sensor boards. Most of the experiments were made with that suit, whose jacket is pictured in Fig. 5.2. Additionally, all the technical components were also tested in workwear of the Johanniter Unfallhilfe, which has similar extensions and is also made by rofa. In both suits, the sensors are completely covered in hook-and-loop tape. The sensors are mounted on the inside of the suit to maintain the suit's safety properties, most importantly heat and fire resistance.

The more technical components are fifteen sensor boards and an embedded PC, all custom made for the suit by Budelmann Elektronik. The heart of the embedded PC, which executes the orien-

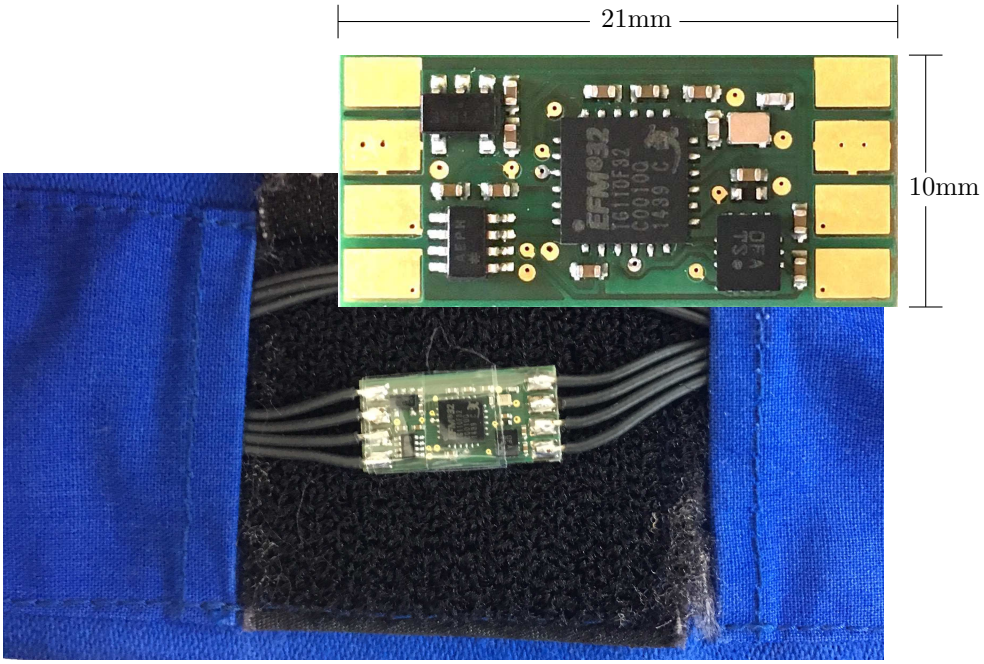


Figure 5.3 *SIRKA sensor board.*
The sensor boards sits approximately in the middle of a piece of hook-and-loop tape (not closed for the picture) on the inside of the suit in an arbitrary orientation. The sensor board itself bears an ARM Cortex-M3 processor and the inertial sensor.

tation estimator, is the single-core ARMv7 processor Freescale i.MX6 using 512MB RAM running Linux 3.18.6 and the embedded Linux distribution Buildroot 2015.02. For communication, the embedded PC also includes an FTDI FT4232H to connect to up to four serial RS485 busses, three of which are used to connect the sensors to the embedded PC. The RS485 busses are operated at 2Mbaud. The measurement models of the orientation estimator are defined for a state which includes data from all sensors up to the same instant. This requires the sensor boards to send data from approximately the same instant. To make them do this, the sensor fusion broadcasts a command to request data to all sensor boards on all interfaces exactly simultaneously with the desired estimation rate.

To prevent the operating system from causing delays between the command on the different interfaces, the embedded PC includes a hardware indirection. Instead of sending the command directly on the serial interfaces sequentially, the sensor fusion configures an additional microcontroller, called *synchronizer*, to periodically send the command to request data. The transmit wire of the synchronizer's serial interface is electrically connected to the three serial busses.

The sensor board, on display in Fig. 5.3, carries a Bosch BMI160 inertial sensor and an ARM Cortex-M3 computer. The sensors are sampled at 200Hz. For both the gyroscope and the accelerometer signals, the inertial sensor has low-pass filters, whose cut-off frequencies are implicitly configured by configuring the sampling rate. At 200Hz the cut-off frequencies of the gyroscope's and the accelerometer's low-pass filters are 74.6Hz and 80Hz, respectively. Both gyroscope and accelerometer provide their measurements as 16-Bit fixed-point values to the on-sensor-board computer. The gyroscope range is configured to $\pm 1000°/s$, that is $1/32.8°/s$/Bit resolution. The

accelerometer range is configured to $\pm 8g$, so its resolution is $1/4096g$/Bit. Bosch provides further details about the sensor in its data sheet [Bos].

The ARM Cortex-M3 is clocked at 32MHz. Inertial sensor and processor are connected via their Serial Peripheral Interface (SPI). The processor does not have a floating-point unit, however, it is able to compute single-precision floating-point numbers via emulation, which is transparently supported by the GCC[1] C compiler.

Fig. 5.4 provides an overview of the components used in a SIRKA suit. The individual sensor boards are per-bus-uniquely addressed. Thus, an individual sensor board is identified by the pair of bus number and sensor address. Access to the bus is time-multiplexed. Since the time slot for a particular sensor board is determined by its address, the addresses on a bus always start at 1 and do not have any gaps.

The postures estimated by the orientation estimator are written to an SD card, from where they can be read using an external computer for evaluation. In fact, the SD card is the only mechanism for the sensor fusion computer to communicate data with the outside, so accumulated sensor data for calibration as well as the calibration results are also written and loaded, respectively, from that SD card. For demonstration purposes, posture estimates can also be communicated via Ethernet, however, in its default configuration the sensor fusion computer does not have an Ethernet port.

[1] I used GCC version 4.8.3 to compile the program for the Cortex-M3.

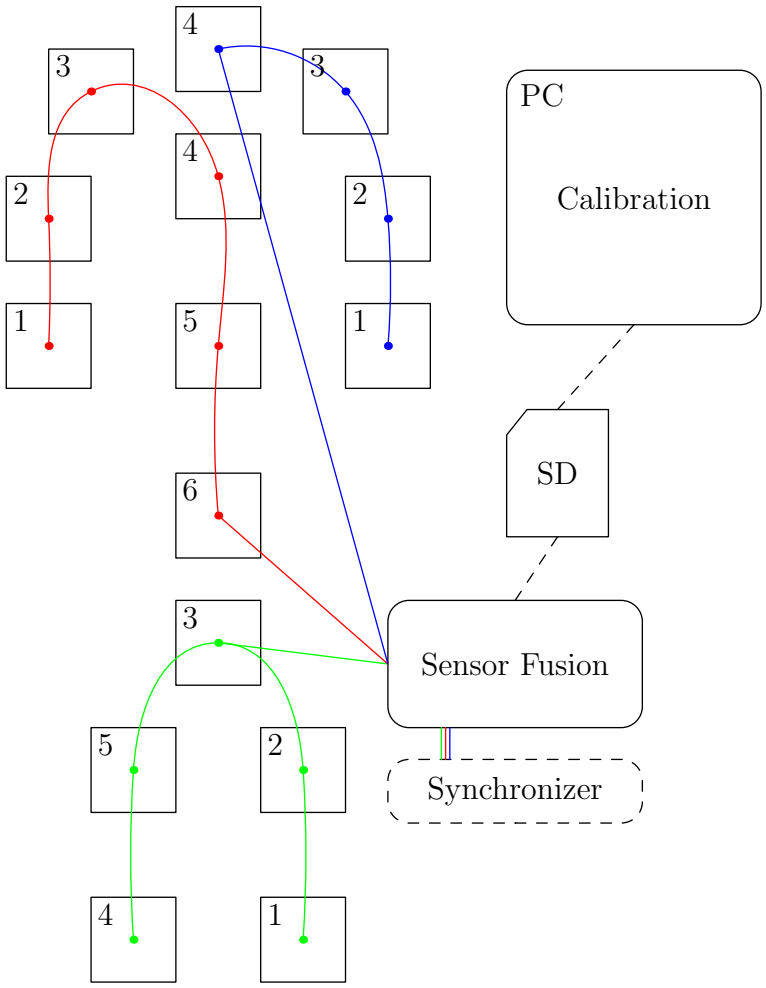


Figure 5.4 *SIRKA System Overview.*
The SIRKA suit is made of 15 sensors (squares), connected using three separate RS485 busses (colored lines). If in accumulation mode (the usual mode), the sensors synchronously communicate accumulated sensor data to the sensor fusion computer. Synchronization of all three busses is maintained using special-purpose hardware integrated into the sensor fusion computer. The sensor and suit calibration is read from an SD card. The calibration itself is computed on an external PC. Except for data exchange over the SD card, the system is entirely self-contained.

5.3 Calibrators

To get the algorithm of Chap. 4 to work in real-time on the hardware described in Sect. 5.2, I developed a number of programs. The posture estimator is split into two essential programs. The SIRKA sensor fusion implements the Extended Kalman Filter to estimate the orientations. Additionally it provides a socket to connect to external software to communicate resulting posture estimates.

The sensor fusion does not process raw sensor data, but the accumulated data instead. The accumulation of inertial sensor data happens in the second essential program, the driver program running on each sensor board.

In addition to the programs performing the primary tasks for the suit I developed two calibrators, which are meant to be run on an external computer. The extrinsics calibrator implements Sect. 4.10 to calibrate the suit on its wearer. The intrinsics calibrator calibrates parameters pertaining the the individual sensors: gyroscope bias as well as accelerometer bias, scaling and misalignment.

Data is read and written by the calibrators from human-readable text files in a format defined by a small library I implemented, called *libimureading*.

The extrinsics calibrator is fed with a time series of accumulates and the corresponding angular velocities for each sensor of the suit and a JSM defining the structure of the suit. The accumulate time-series is obtained by a small companion program which runs on the sensor-fusion computer. Instead of estimating postures, it writes the sensor data accumulates sent by the sensor boards into text files onto the SD card. The extrinsics calibrator computes for each sensor the relative accumulates and sets up SLOM [Her08], the framework for least-squares problems involving manifolds, to solve the calibration problem according to Sect. 4.10.

The data required to calibrate the skeleton is not completely arbitrary. Every DOF of the suit wearer, as defined by the skeleton structure in Sect. 5.1, should be moved. To enable the calibrator to determine the orientations of the sensors, the suit wearer as a whole also has to accelerate, e. g. walk and stop abruptly. In addition to the calibration software, I tested a "calibration exercise", which was video-taped at one of the evaluation occasions as a reference.

The calibration exercise begins with standing upright for the first 10 seconds, with the relaxed arms hanging downwards. I assume this posture to be the approximate posture of the suit wearer when the suit starts normal operation. Thus, the orientations of the bodies after 5 seconds are written to text files, which are to be loaded by the sensor fusion and serve as the initialization for the orientation estimates.

Both the extrinsics calibrator and the Kalman Filter to estimate orientations use the sensors's coordinate systems only. In the measurement models, no separate body coordinate system comes up, everything is expressed in sensor-coordinates. So the coordinate system of a body is defined by the axes of the sensor mounted on the body. However, there also is a physical definition of the body-coordinate systems. These are the coordinate systems the bodies in Fig. 5.1 are annotated with. At the beginning of the calibration exercise, in the initial, relaxed posture, the physical orientations of the bodies in world coordinates are approximately known. Thus, the initial sensor

orientations from the calibration exercise are also used to determine the approximately constant orientation offsets between sensor coordinates and physical coordinates of each body.

To properly accumulate inertial sensor data, each sensor board needs to know the intrinsic calibration of its inertial sensor. These include the gyroscope bias, $b_g \in \mathbb{R}^3$, the accelerometer bias $b_a \in \mathbb{R}^3$ and the accelerometer gain matrix $A \in \mathbb{R}^{3\times3}$. The accelerometer gain combines the rotation offset between gyroscope and accelerometer, $Q_{g\curvearrowleft a} \in \mathbb{SO}(3)$, the non-perpendicularity matrix P, which maps the coordinate system spanned by the not necessarily perpendicular sensor axes to a coordinate system with orthogonal axes, and the scaling S, a diagonal matrix with the scale factors of each sensor axis. Thus $A = Q_{g\curvearrowleft a}PS$. With the true acceleration $\hat{a}$, the accelerometer measurement a, subject to zero-mean, Gaussian measurement noise with covariance Σ_a, is

$$a = A^{-1}\hat{a} + b_a + \epsilon \quad \epsilon \sim \mathcal{N}(0, \Sigma_a) \quad . \tag{5.1}$$

Before the sensor boards are mounted inside the suit, they are hooked up to an external computer and mounted on an approximately cuboid block. Again using a simple companion program, raw data of the sensors mounted on the block is recorded. Initially, block and sensors are at rest on one of the block's sides for a little more than $T_{\text{rest}} = 20s$. Afterwards the block is put on each of its other sides for about $10s$. While N timestamped inertial measurements are recorded using this procedure, the accelerometer measures negative gravity in three almost perpendicular directions with both possible signs. For the k'th measurement, timestamp, gyroscope measurement and accelerometer measurement are t_k, ω_k and a_k, respectively. Timestamps are normalized such that $t_1 = 0$.

In the first $T_{\text{rest}} = 20s$, the sensors are at rest, so the initial guess for the gyroscope bias, $\bar{b}_g$ is

$$\bar{b}_g = \sum_{k=1}^{N_{\text{rest}}} \frac{\omega_k}{N_{\text{rest}}} \quad \text{with } N_{\text{rest}} = \max\{k \mid 1 \le k \le N, t_k \le T_{\text{rest}}\} \quad . \tag{5.2}$$

The initial orientation of the sensor is determined from the first accelerometer measurement a_1 using the solution eq. (2.22) to Wahba's problem, eq. (2.21):

$$Q_{W\curvearrowleft 1} = U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det U \det V \end{bmatrix} V \text{ with } USV = \text{SVD}\left(-g a_1^T\right) \quad , \tag{5.3}$$

where g is the usual gravity vector. This is analogous to eq. (4.103) to find initial sensor orientation in the extrinsics or skeleton calibration problem. From there, again analogous to the skeleton calibration, the initial guess of the sensor's subsequent orientations is computed recursively:

$$Q_{W\curvearrowleft k} = Q_{W\curvearrowleft k-1}\,\text{Rot}\left(\left(\omega_k - \bar{b}_g\right)(t_k - t_{k-1})\right) \quad \text{for } 2 \le k \le N \quad . \tag{5.4}$$

If the angular speed is less than $\psi_{\text{rest}} = 1°/s$, I assume the sensor to be at rest. Thus, its accelerometer should measure negative gravity. The set of indices of measurements taken at rest is $\Omega_{\text{rest}} = \left\{k \mid 1 \le k \le N, \|\omega_k - \bar{b}_g\| \le \psi_{\text{rest}}\right\}$. All this gives rise to the following least-squares

problem, which is solved by the intrinsics calibrator using, again, SLOM:

$$[b_g, b_a, A] = \arg \min_{[Q_{W\curvearrowleft k}]_{k=1}^{N}, b_g, b_a, A} \frac{1}{2} \left\| \begin{bmatrix} \left[\omega_k - b_g - \mathrm{aRot}\left(Q_{W\curvearrowleft k-1}^{-1} Q_{W\curvearrowleft k}\right)/(t_k - t_{k-1})\right]_{k=2}^{N} \\ \left[\omega_k - b_g\right]_{k=1}^{N_{\mathrm{rest}}} \\ \left[a_j - b_a - A^{-1} Q_{W\curvearrowleft j}^{-1}(-g)\right]_{j \in \Omega_{\mathrm{rest}}} \end{bmatrix} \right\|^2 . \tag{5.5}$$

The initial guess is zero for the accelerometer bias and the identity for the accelerometer gain matrix. To obtain a better initial guess for the accelerometer calibration, an ellipsoid could be fitted to the accelerometer measurements first. However, accelerometer measurements are almost spherical, i. e. A stays close to the identity. For magnetometers, which have a very similar measurement model — measuring the magnetic instead of the gravitational field — the preliminary ellipsoid fitting is quite common, for instance as described by Kok et al. [Kok+12], since the measurements obtained by rotating the uncalibrated sensor in the field lie on a more eccentric ellipsoid.

The intrinsics calibration process is repeated for each sensor. When operating the sensor to accumulate inertial sensor data, copying the intrinsics calibration to the sensor is the most important part of the sensor initialization.

5.4 Sensor Board Program

The program running on the sensor board sets the sensor up, queries the sensor for new sensor data, applies the intrinsics calibration to the sensor data and accumulates the sensor data. Additionally, it has to communicate with the central sensor fusion computer without disturbing the other sensor boards, which are connected on the same bus.

The communication with the sensor itself over SPI was implemented by Budelmann Elektronik, as was the initialization of the Cortex-M3 itself. My contribution to the sensor board software begins after the clocks and the RS485 bus are already set up. The Cortex-M3 supports the Cortex Microcontroller Software Interface Standard (CMSIS) [ARM], which I use particularly for floating-point and linear algebra calculations as well as to look up sines and cosines in pre-computed tables. To accumulate sensor data, the sensor board needs to accumulate orientations and rotate 3-dimensional vectors. To do this, I implemented a small library based on the CMSIS linear algebra functions, which represents orientations as unit quaternions. The library efficiently rotates vectors using a unit quaternion, creates unit quaternions from 3-dimensional, scaled-axis rotation vectors, and multiplies and normalizes quaternions.

After configuring sampling rate and measurement range by setting the sensor's corresponding registers over SPI, the sensor board driver runs the infinite loop displayed in Fig. 5.5. The loop sequentially processes sensor data and communicates with the central sensor fusion computer by performing the following tasks.

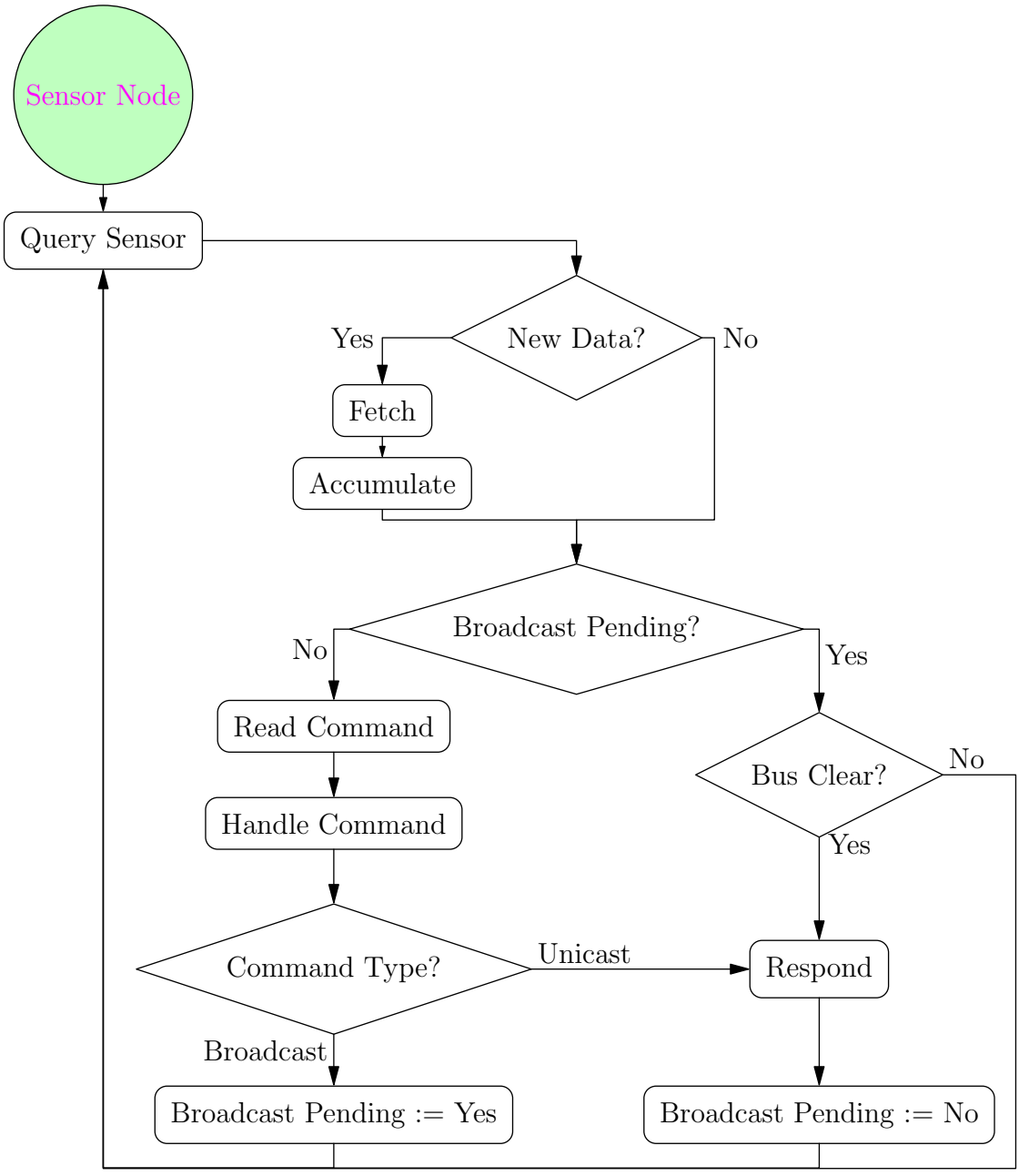


Figure 5.5 *SIRKA sensor node controller program flow.*
As long as the sensor node has power, its microcontroller runs an infinite loop of sensor data processing and communication on the sensor network. The program contains no waiting, allowing the controller to sample the sensor with 200Hz.

Processing Sensor Data

The sensor board program maintains the time in microseconds between initialization and last data accumulation, t, as a 32-Bit unsigned integer and both the accumulated sensor data,

$$M = \begin{bmatrix} Q & v \\ 0 & 1 \end{bmatrix} \ , \tag{5.6}$$

as well the last gyroscope and accelerometer reading obtained from the sensor, ω_{raw} and a_{raw}, respectively. Readings from the inertial sensor are timestamped by the sensor itself, using an unsigned 24-Bit integer with $39\mu s$ per Bit resolution. This raw timestamp t_{raw} is retained separately. t_{raw} is initialized by fetching a time-stamped measurement, whose measurements are otherwise not used.

The accumulate is initialized to the identity orientation and zero velocity and is stored as single-precision floating-point values. The sensor data is stored as encoded by the sensor, i. e. as 16-Bit signed integers. t flows over after about 71.5 minutes. Both the overflow and the initialization are not of practical importance, because the Extended Kalman Filter uses only relative accumulates. The overflow in t is also not likely to be the first overflow to occur. Each component of the velocity vector flows over at $\pm 50 m/s$. Assuming a worst-case acceleration of $10g$, the second wrap-around of a velocity component might occur after about a second. Thus, to ensure that the central sensor fusion can reconstruct the relative velocity, two subsequent accumulates sent to the central sensor fusion might not be further than a second apart. However, this appears to be a rather long time given the estimation frequency of 10Hz. Making the components of v wrap around avoids saturation to the floating-point infinity value and also keeps the floating-point accuracy reasonable.

Query Sensor The inertial sensor indicates via a flag in one of its registers whether or not new data is available. The sensor board program checks this flag before updating any of its data structures.

Fetch If new data is available, the raw data is copied from the sensor. Using the temporary variable, $t'_{\text{raw}} := t_{\text{raw}}$, to retain the previous raw timestamp, $t_{\text{raw}}, \omega_{\text{raw}}$ and a_{raw} are updated to the most recent values provides by the sensor. $:=$ denotes the assignment operation.

From the updated raw timestamp, the corresponding sampling duration is $\Delta t = 39\mu s(t_{\text{raw}} - t'_{\text{raw}})$.

Accumulate The newly fetched sensor data is scaled using the resolutions of gyroscope and accelerometer, $\eta_\omega = (1/32.8)°/s$ and $\eta_a = (1/4096)\|g\| m/s^2$, and calibrated using the intrinsic parameters, to obtain the current angular velocity ω and the acceleration a:

$$\omega = \eta_\omega \omega_{\text{raw}} - b_g \quad a = A(\eta_a a_{\text{raw}} - b_a) \tag{5.7}$$

These are used to compute the incremental accumulate as in eq. (4.43), which is added to total accumulate

$$M = \begin{bmatrix} Q & v \\ 0 & 1 \end{bmatrix} := \begin{bmatrix} Q\,\mathrm{Rot}(\omega\Delta t) & \mathit{oflow}\,(v + Q\,\mathrm{Rot}(\omega\Delta t)a\Delta t) \\ 0 & 1 \end{bmatrix} \tag{5.8}$$

$$\mathit{oflow}(v) = [\mathit{cflow}(v_k)]_{k=1}^{3} \quad \text{with } \mathit{cflow}(x) = \begin{cases} x' - 100 & \text{if } x' \geq 50 \\ x' + 100 & \text{if } x' < -50 \\ x' & \text{otherwise} \end{cases} \text{ and } x' = x \mod 100 \tag{5.9}$$

$(x \mod y)$ is the floating point remainder, that is $(x \mod y) = x - k * y$ such that k is an integer and the result has the same sign as x [KR88, chapter B4]. The accumulation time $t := t + \Delta t$ is incremented accordingly. The *oflow* function handles the overflow of the velocity component of the accumulate explicitly.

Communications

The main requirements for the communications part are to avoid disturbing the communications of other sensor boards on the same bus and not to busy-wait. That is, whenever it wants to send data to the sensor fusion computer, the sensor board has to wait for the bus to become clear. Waiting must not happen in a loop which would prevent the inertial sensor from being queried, leading to data loss.

The sensor board has to deal with two kinds of messages: Messages addressed directly to itself and broadcast messages, which are processed by all sensor boards on the bus. Non-broadcast (or "unicast") messages are sent by the central sensor fusion computer during setup, broadcast messages during normal operation, i. e. it is save for the sensor board to consider the bus to be clear directly after a non-broadcast message has been received. Thus, responses to non-broadcast messages do not have to be synchronized with other sensor boards.

The sensor board handles commands sequentially; no other message is received between receiving a message and sending the corresponding response.

Two buffers are maintained for communications, a receive-buffer for incoming messages and send-buffer for outgoing messages.

Read Command The receive-buffer is filled concurrently by the interrupt handler triggered by incoming data on the serial bus. Before a command is handled, the current contents of the receive-buffer are copied.

Handle Command Only commands addressed to the specific sensor board or broadcast messages are considered. Broadcast messages are identified by the number 128 in the address field of the message. The important non-broadcast messages are a query for the program version, a configuration message to change the sensor board's address, and a message to communicate the

intrinsics calibration of the inertial sensor. All messages are subject to a 16-Bit cyclic redundancy check (CRC) and are responded to immediately, at least with a success-or-failure indicator if no data has to be responded. An exception is the message to update the sensor board software. This message is processed by configuring the sensor board to boot into a program supplied by Budelmann Elektronik, which accepts a new sensor board binary and configures the sensor board to boot the updated software afterwards.

There are two broadcast messages, which are responded to by either sending the current accumulate, angular velocity and timestamp, or by sending the most recent raw sensor data. Since all sensor boards have to respond to the broadcast message, the response is not sent immediately. Instead, the complete response message is assembled in the send-buffer, the Broadcast-Pending flag is set and the Bus-Clear flag is cleared, if the sensor board's address is bigger than 1. Additionally, for all sensors but the first on the bus, a hardware timer is set to approximately $(\text{address} - 1)5ms$, which triggers a hardware interrupt whose handler sets the Bus-Clear flag. Thus, the first sensor on a bus sends data immediately, while other sensors wait for their time slot concurrently to the main program flow, i. e. concurrently to accumulating sensor data.

Respond The actual sending is just writing the send-buffer on the serial bus.

Doing so ensures that no data is lost while waiting and that the responses, although sent sequentially, all contain data from approximately the same instant.

In retrospect, getting the communications right was as laborious as implementing the data processing and accumulation components.

5.5 Sensor Fusion Program

The program on the embedded PC sets up and receives accumulated sensor data from the sensor boards. It implements the Extended Kalman Filter to estimate orientations and writes the resulting posture estimates to a file and, if a client program is connected, to a TCP socket.

At program start, the intrinsics calibration and the JSM with the calibrated skeleton structure are loaded. Additionally, the orientations of all sensors from 5 seconds into the calibration exercise are loaded, which are used to initialize the state of the orientation estimator. Initial velocities and biases are zero.

At 5 seconds into the calibration exercise, the wearer of the suit stood approximately upright, such that both the physical orientations and the sensor orientations of his bodies are approximately known.

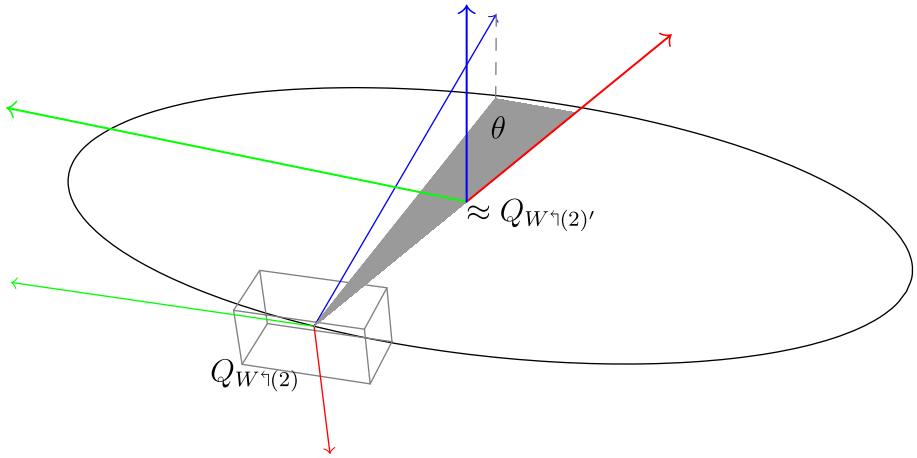


Figure 5.6 *Body-to-Sensor-Orientation*
The grey box denotes the sensor mounted on the back of body 2, whose contour is denoted by the ellipse. x-axes are colored red, y-axes green and z-axes blue. The z-axis of the sensor coordinate frame of body 2 points approximately in the direction of its physical x-axis, which in the default posture is the same as the world-x-axis except for an angle θ in the x-y-plane, which has to be accounted for before computing the orientation offset between physical and sensor coordinates.

Sensor-Orientation to Physical Orientation

Because the sensors are mounted in the suit in an arbitrary orientation, the physical orientations rather than the sensor orientations are the final output of the sensor fusion program. The orientation offset between the sensors coordinate system and the physical coordinate system of each body stays approximately constant and is determined as follows.

The heading of the skeleton as a whole, called the total heading, is arbitrary. Thus, the total heading should not influence the orientation offsets between sensor and physical coordinates. For the purpose of finding the orientation offsets, I define the total heading of the skeleton to be the heading of the lowest body of the torso, i. e. body 2 in Fig. 5.1. This is convenient, because the sensor on this body is mounted inside the waistband under the belt. This construction makes the z-axis of the sensor coordinate system approximately point in the direction of the x-axis of the physical coordinate system, which approximately coincides with the world coordinate system in the initial posture. In this situation, drawn in Fig. 5.6, the total heading is the angle θ the projection of the sensor-z-axis onto the ground-parallel plane makes with the world-x-axis.

The sensor-z-axis in world coordinates is the third column of the sensor orientation. So the total heading expressed as a rotation matrix $Q_{W\urcorner W'}$ is

$$Q_{W\urcorner W'} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ with } \theta = \operatorname{atan2}(x, y) \text{ and } \begin{bmatrix} x \\ y \\ z \end{bmatrix} = Q_{W\urcorner(2)} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} . \tag{5.10}$$

W' denotes the world coordinate system rotated around the vertical axis to match the global

heading θ. This is the coordinate system I interpret the initial physical orientations of the bodies in. In Fig. 5.1 the initial posture of the skeleton is defined with all the body-x-axes pointing forward (into the paper). Since for the posture the total heading does not matter, forward in the direction with heading θ is as good as any other forward direction, as long as the direction is in the ground-parallel plane.

So for each body k, the initial orientation can be read off of Fig. 5.1 to create their rotation matrix representations $Q_{W'\neg(k')}$.

The orientation offset between sensor coordinates and physical coordinates for body k is thus

$$Q_{(k)\neg(k')} = Q^T_{W\neg(k)} Q_{W\neg W'} Q_{W'\neg(k')} \ , \tag{5.11}$$

where $Q_{W\neg(k)}$ is the orientation of body k in world coordinates 5 seconds in to the calibration exercise as loaded from the calibrator output.

Posture Reconstruction

With the orientation offsets between sensor coordinates and physical coordinates of the bodies known, a physically meaningful posture reconstruction can be built by joining the bodies in their physical orientations.

Using a table of vectors specifying the location of the origin of each body in physical coordinates of its predecessor body, that is for body k $r_{(\lambda(k)')\neg(k')}$, the posture, that is all *poses* as opposed to orientations only, is computed from the orientation estimates $Q_{W\neg(k)}$ recursively:

$$T_{W\neg(k')} = \begin{bmatrix} Q_{W\neg(k')} & r_{W\neg(k')} \\ 0 & 1 \end{bmatrix} \tag{5.12}$$
$$\text{with } Q_{W\neg(k')} = Q_{W\neg(k)} Q_{(k)\neg(k')} \text{ and } r_{W\neg(k')} = r_{W\neg(\lambda(k)')} + Q_{W\neg(\lambda(k)')} r_{(\lambda(k)')\neg(k')} \ .$$

So the poses are constructed from the orientation estimates and the skeleton structure by traversing the kinematic tree.

Orientation Estimation

Except for the transformation from sensor coordinates to physical coordinates, most of the sensor fusion program implements the Kalman Filter to estimate the orientations from accumulated inertial sensor data and interfaces with the sensor boards. The main steps following loading the calibration and determining the coordinate system offsets are displayed in Fig. 5.7.

Check Sensors Initially, the presence of each sensor on each bus is checked. If there are sensors missing, the program is aborted. The test is carried out by asking each sensor board for its software version.

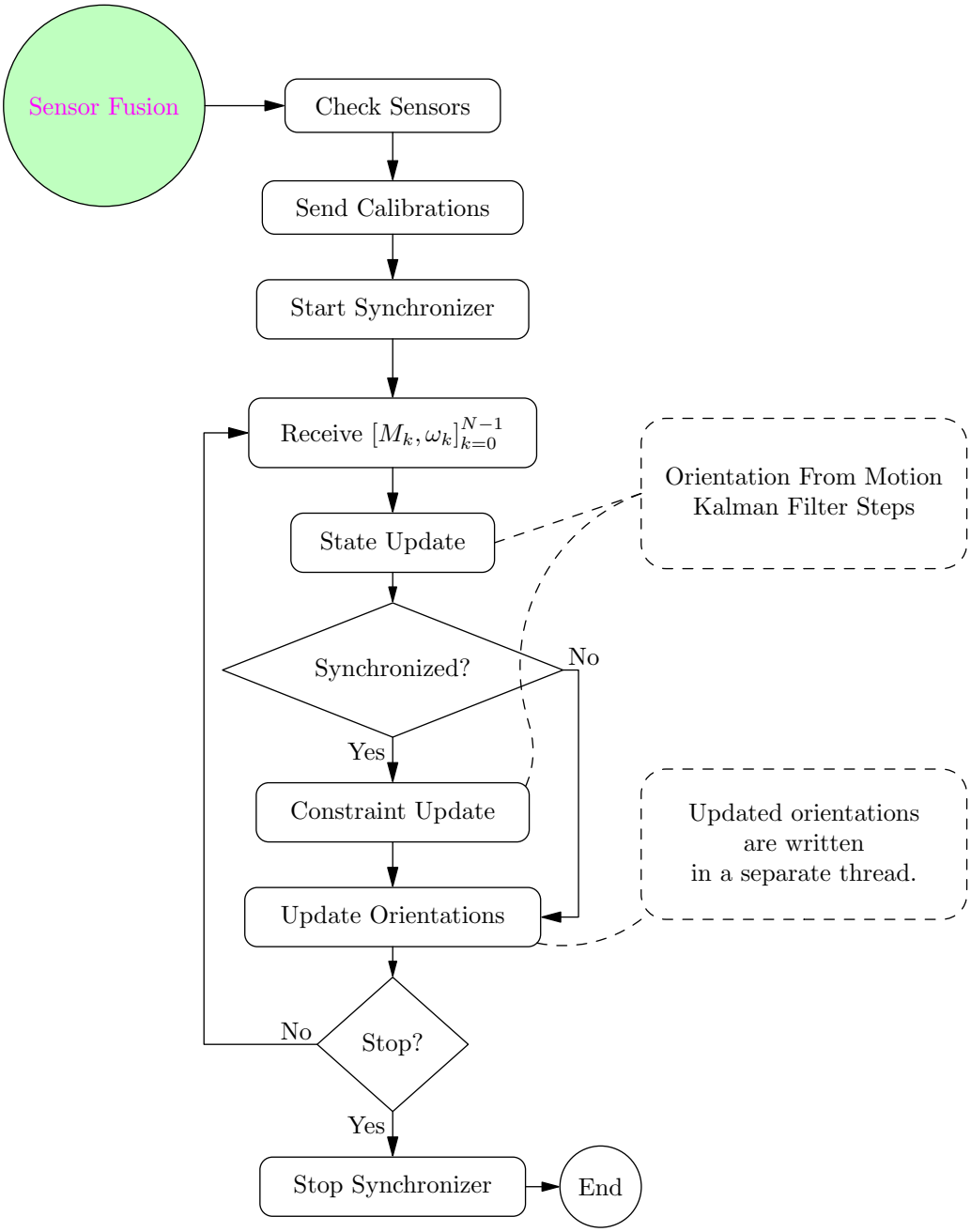


Figure 5.7 *SIRKA sensor fusion program flow.*
The orientation from motion estimator is fed with relative relative orientations and velocities computed from the accumulated data received from the sensors. The correction step based on the skeleton's structure only happens if synchronized data has been received from *all* sensors.

Send Calibrations If all 15 sensors are present, the loaded intrinsics calibration is sent to each sensor sequentially. The last calibration message is also the last non-broadcast message sent by the program.

Start Synchronizer Using a tiny library I wrote for the synchronizer, the synchronizer is configured to send the broadcast query for accumulated sensor data every $100ms$ on all busses at once.

Receive $[M_k, \omega_k]_{k=0}^{N-1}$ From here on, the sensor fusion program only receives data over the three serial interfaces. Since the synchronizer is electrically connected to all three busses, the sensor fusion program waits for the broadcasted data query to appear on all busses and, following the broadcast, for the responses of all sensors boards.

State Update The components of the estimator state corresponding to all sensors that sent data are updated. This implements the dynamics update of the Kalman Filter as described in Sect. 4.5 and 4.7.

Synchronized? The sensor fusion program keeps track of which sensor board responded to the most-recent query for sensor data. A sensor board did not respond if either it did not send anything or if the response failed the CRC.

If the response of at least one sensor board is missing, the sensors are deemed not to be synchronized. Let $\delta T^{(k)}$ be the relative accumulation period of the sensor board of body k since the accumulate sent in the previous response of body k. All relative accumulate times are thus $Ts = \{T^{(k)} \mid 1 \leq k \leq 15\}$. Even if all sensor boards responded, but $\max(Ts) - \min(Ts) \geq 5ms$, the sensor boards are also deemed to be unsynchronized.

Constraint Update If the sensors are synchronized, which is the case most of the time, the (pseudo-)measurement update of the Kalman Filter is executed as described in Sect. 4.5 using the joint models from Sect. 4.6.

Update Orientations Finally the orientations of the bodies are copied into a separate orientations array. This separate orientations array is shared with a *writer* thread, which executes concurrently to the estimation loop described here and displayed in Fig. 5.7.

The writer thread carries out the posture reconstruction in physical coordinates described above. The results in physical coordinates are then written to a file on the SD card and optionally transmitted over the TCP socket. Data to write or send is prepared with Google's Protobuf library using a specification the SIRKA project partners came up with in the beginning of the project.

Data is sent or written concurrently because the time it takes to write and send data appears to be unpredictable and the sensor fusion program must keep up with the 10Hz estimation rate. This is usually not a problem, the platform is powerful enough to estimate with 20Hz, however, a stalled socket would slow things down too much.

Stop? The program implements a rudimentary button interface. The sensor fusion computer has two buttons connected via General-purpose input/output (GPIO). If either button is pressed, the sensor fusion program stops. One of the buttons, called the "start/stop" button, lets the sensor fusion program restart and write the new posture data to a newly created output file. The other button, "shutdown", lets the sensor fusion program set up a script which unmounts the SD card an shuts down the sensor fusion computer.

Stop Synchronizer Before the program exits, the synchronizer is configured to stop querying the sensors.

5.6 Miscellaneous Tools

More small programs exist to maintain the sensor network. There is an Updater, which has been mostly developed at Budelmann Elektronik, to copy new sensor board program binaries to the sensor boards.

I developed a program to change the address of a sensor board, as well as a program to receive raw sensor data from all sensors on a single bus. Both programs are meant to be run on a PC (as opposed to the embedded PC for the sensor fusion) to configure and debug individual busses. The recorder for raw data optionally plots the norms of the measurements of all sensors on a bus in a single plot. This is particularly handy to determine which sensor board has which address. Once the sensor boards are soldered together, individual sensor boards can not be disconnected, so the only chance to debug a potentially messed up sequence of sensors is to move individual sensors while looking at the stream of data.

Also very useful for debugging and, as it turned out at various fairs and presentations, for demos is a simple network client, which reads the stream of posture data in real-time.

5.7 Simulation: Calibrating a Skeleton from Noisy Data

From Sect. 4.8 it is already known that the orientation estimator works in a rather short-term experiment when sensors are mounted directly on rigid bodies. Even when accumulated instead of raw sensor data is used to estimate orientations, the relative orientations are recovered accurately.

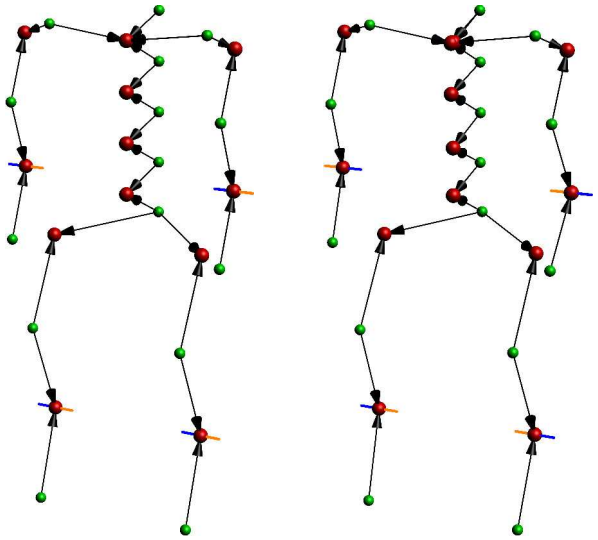


Figure 5.8 *Skeleton Calibration vs. Ground Truth*
Visualizations of the JSM of the SIRKA skeleton approximation. The left shows the ground truth used to generate sensor data. The right skeleton visualizes the result of the calibrator. Red spheres denote joints, green spheres denote sensors. Arrows depict the joint locations. Hinges are drawn in orange and blue, where orange denotes the axes in coordinates of the preceding sensor, blue axes in coordinates of the succeeding sensor.

What remains to be seen is how the estimator behaves in a long-term experiment. Moreover, in the targeted application for the suit, sensors are not mounted perfectly rigidly on perfectly rigid bodies. Instead, the sensors are mounted in cloths worn by a worker, so the really interesting test is whether posture estimates obtained from a person wearing the suit with integrated sensors look plausibly like the actual posture of the person.

Before any postures can be estimated, the calibrator has to provide the joint locations and hinge axes of the skeleton. To test the calibrator, I tried to reproduce the joint locations and hinge axes of a simulated skeleton, structured as described in Sect. 5.1.

Each DOF of the simulated skeleton was moved one after another. To provide information about relative orientations around the world-vertical axis, the skeleton as a whole was moved once along the world-x-axis, i. e. all accelerometers measured an acceleration different from gravity. The standard deviations of gyroscope noise and accelerometer noise were $(0.007\pi/180 * \sqrt{200})/s$ and $(300 * 10^{-6} * 9.81 * \sqrt{200})m/s^2$ respectively. To provide reasonably realistic values, I interpreted both values from the documentation [Bos] of the sensor.

The average difference between the calibrated and true joint locations was 11.1mm. The biggest difference was 25.3mm. The hinge axes were recovered almost exactly. The worst (angular) difference was 0.4°, the average difference is 0.13°.

The calibrated skeleton is visually almost indistinguishable from the visualization of ground truth, as shown in Fig. 5.8.

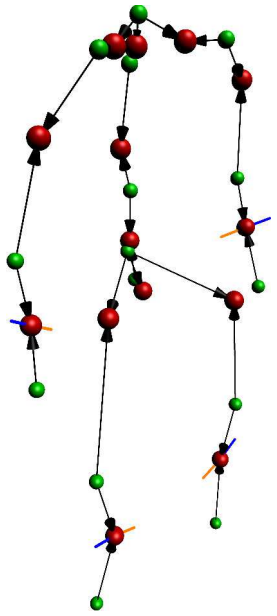


Figure 5.9 *Sensor Suit Skeleton Calibration Result*
Red spheres denote joints, green spheres denote sensors. The visualization's color-coded "syntax" is the same as in Fig. 5.8. The result is not nearly as good as on generated data. However, legs and arms appear to be properly reproduced and the torso as a whole is also not entirely implausible.

5.8 Calibrating a Suit from real Sensor Data

To operate a real suit, the real suit has to be calibrated. Aside from sensor noise, other effects worsen the result. These include effects pertaining to individual sensors, most importantly probably slight miscalibration. Larger errors are introduced by the deliberately false model: the hinges in the model are not perfect hinges in reality, the real spine also has a lot more than 5 bodies, the sensors are not mounted perfectly rigidly but move with the cloths, etc.

It is also hard to move each DOF fast enough to make the small displacements between joints and sensors cause a sufficient effect. In particular this affects the spine, where the model is presumably worse than at the other portions of the skeleton.

However, the orientation estimator uses the same models as the calibrator. Thus, displacements, including their errors, that do not have a large effect in the calibrator, do not have a large effect in the orientation estimator either.

The result of the calibration run is rendered in Fig. 5.9. The calibration result appears to be plausible for the arms and legs, i. e. for the bodies with rather large displacements between joints and sensors. The spine at least looks upright, although the joint positions are just the least-squares fit of where the joints would be if the 5-body-spine-model were true. Note that the heading of the calibrated skeleton as a whole is arbitrary, as it is in the posture estimated by the orientation estimator. The calibration exercise used to obtain this calibration was performed by

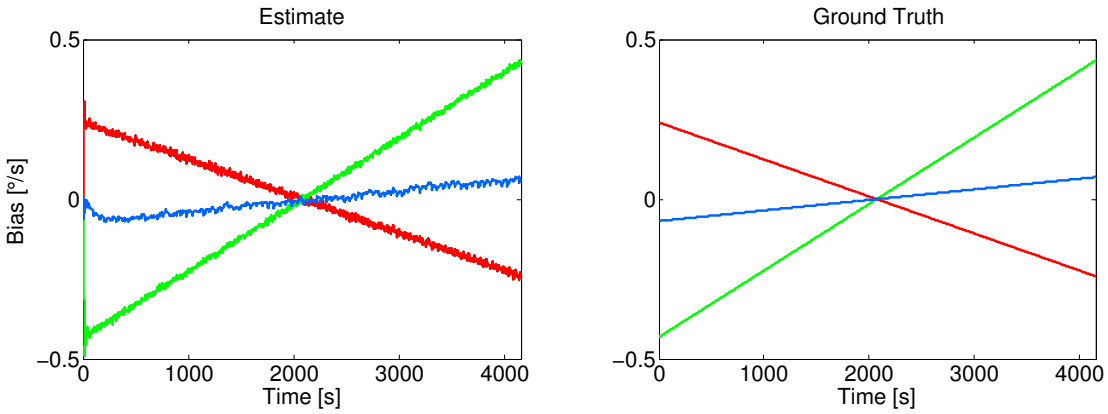


Figure 5.10 *Sirka Simulation: Gyroscope Bias*
The estimate (left) and simulation (ground truth) of the bias of the gyroscope of sensor 0 over the simulation. The simulated bias changes linearly over $1°/s$. Except for the very beginning, the gyroscope bias is tracked accurately.

one of the Meyer Werft's workers and was recorded in a video[2].

5.9 Simulation: Long Term Posture Estimation from Noisy Data

To test the estimator's long-term behavior and to quantitatively assess the estimation error, I simulated the skeleton with a little more than an hour worth of data. The errors of interest are the errors in the relative orientations of adjacent bodies with respect to the corresponding relative orientations obtained from ground truth, and additionally the inclination errors of the skeleton's bodies. The changing biases should also be tracked. Gyroscope biases are approximately constant over short periods, so a longer simulation is the best chance to observe the estimator's behavior regarding the bias.

The simulated sensor data was, as in the calibration experiment of Sect. 5.7, distorted by Gaussian noise with standard deviations of gyroscope noise and accelerometer noise again $(0.007\pi/180 * \sqrt{200})/s$ and $(300 * 10^{-6} * 9.81 * \sqrt{200})m/s^2$ respectively.

Gyroscope data was additionally distorted by a bias. For each gyroscope, a unit vector in a random direction, v, was drawn and the corresponding gyroscope bias changed linearly over the experiment duration from $v0.5°/s$ to $-v0.5°/s$. As an example, the simulated bias of the gyroscope on body 0 is plotted in Fig. 5.10.

The norms of errors of all relative orientations over all joints are plotted in Fig. 5.11. The errors are computed analogously to eq. (4.77). For a joint j, the orientations of the preceding and

[2] *http://www.informatik.uni-bremen.de/agebv2/downloads/videos/wenk_diss/calibexercise.mp4*

succeeding bodies in the estimator state are $Q_{W\curvearrowleft\lambda(j)}$ and $Q_{W\curvearrowleft\mu(j)}$. The corresponding ground truth orientations of the simulated data are $Q^{(\mathrm{gt})}_{W\curvearrowleft\lambda(j)}$ and $Q^{(\mathrm{gt})}_{W\curvearrowleft\mu(j)}$. The resulting norm of the relative orientation error over joint j is thus

$$e_j = \left\| \mathrm{aRot}\left(\left[Q^{(\mathrm{gt})^T}_{W\curvearrowleft\lambda(k)} Q^{(\mathrm{gt})}_{W\curvearrowleft\mu(j)} \right]^T \left[Q^T_{W\curvearrowleft\lambda(k)} Q_{W\curvearrowleft\mu(j)} \right] \right) \right\|_2 \quad . \tag{5.13}$$

Initially, the estimator builds up large orientation errors up to 40°, because the gyroscope biases in the estimator states are all initialized to zero. Hence, in the very beginning, the gyroscope biases are not compensated for. After about 2 minutes, the worst error in any relative orientation is clearly less than 10°, after another minute below 5°.

Despite the large error in the beginning, the average absolute orientation error is only 1.04°. The average absolute orientation error drops to 0.88°, if the first three minutes are discarded.

The graph of the maximum error in Fig. 5.11 displays a rather curious pattern. About every minute, the orientation error drops a little. This is due to the pattern used to generate the test data. The data are simply repetitions of the data used to test the calibrator in Sect. 5.7. So about every minute, the entire skeleton is moved along the world-x-axis, as plotted in Fig. 5.12. For the estimator, this movement is the main source of information regarding relative orientations around the world-vertical axis, because all accelerometers measure an acceleration different from gravity at the same time. So as the estimator gets new information, it corrects the estimate.

In contrast to the orientation components around the world-vertical axis, the remaining two DOF of each (relative) orientation are almost always observable, because gravity acts on each sensor all the time. Accordingly, the inclination of each sensor suffers a lot less from the initially wrong gyroscope biases. The inclination errors, plotted in Fig. 5.13, are very small. The inclination error is calculated by determining the absolute angle between the world-vertical axis in estimated and ground truth body-coordinates, i. e. for body k

$$\phi_k = \angle\left(Q^{(\mathrm{gt})^T}_{W\curvearrowleft(k)} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T, Q^T_{W\curvearrowleft(k)} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \right) \quad . \tag{5.14}$$

In spite of the permanent observability of the inclination, the inclination error also appears to be influenced by the translational movement pattern of the skeleton as a whole. The inclination of each body is estimated mainly using the zero-velocity prior described in Sect. 4.5, which is, of course, affected over periods with a non-zero velocity. However, the worst inclination error stays below 0.5°, i. e. the error is barely noticeable.

In order to sustain an approximately correct estimate, the estimator needs to compensate for the gyroscope biases and thus needs to estimate them, too. The norms of the errors of the estimated gyroscope biases with respect to the simulated gyroscope biases are plotted in Fig. 5.14. For each gyroscope, the initial error is the norm of the simulated bias. Subsequently, the gyroscope bias estimates improve and fall below $0.1°/s$ very quickly. Alongside the example of a generated gyroscope bias used in the simulation, the corresponding estimate is plotted in Fig. 5.10, where the gyroscope bias trajectory is tracked by the estimator. The average absolute error in all estimated gyroscope biases was $0.014°/s$ over the entire simulation.

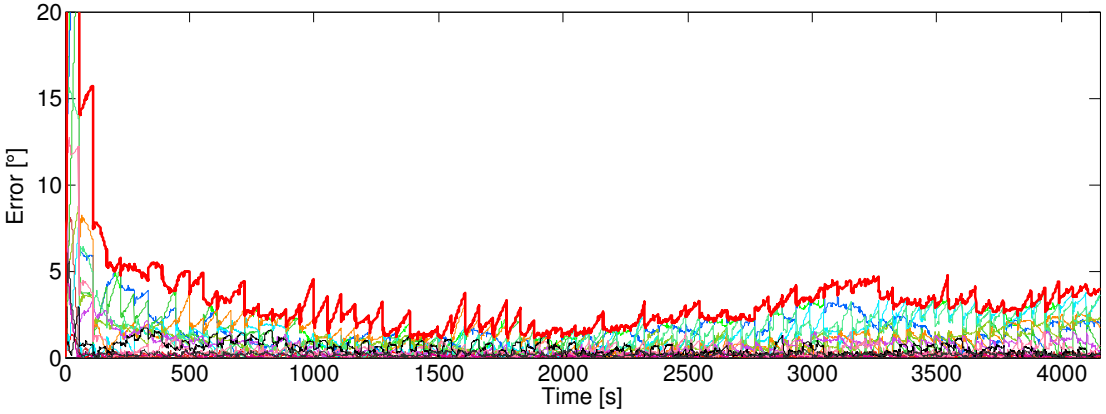


Figure 5.11 *Long Term Simulation Orientation Error*
Norms of errors e_j of all relative orientations of a simulated skeleton are plotted as thin lines. They are computed using eq. (5.13). The important thick, red line traces the maximum of all errors, i. e. $\max\{e_j \mid 1 \leq j \leq M\}$, with the number of joints M. The worst error quickly falls below 5° and stays there.

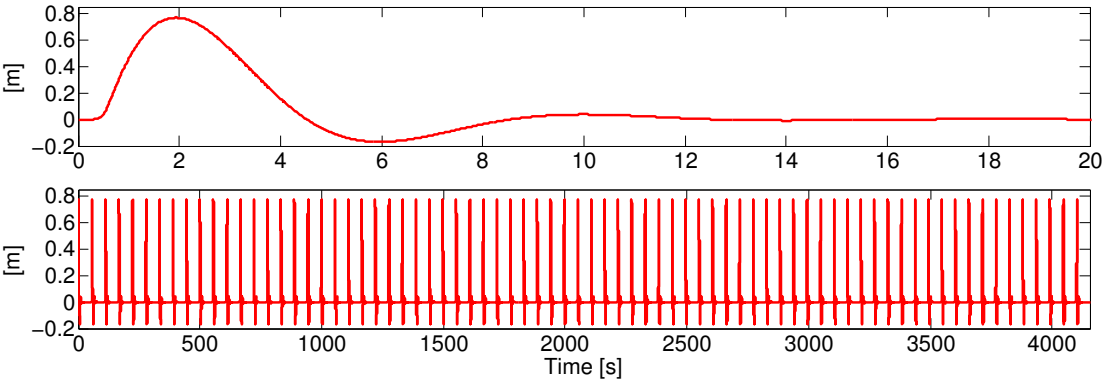


Figure 5.12 *Long Term Simulation Skeleton Position*
The position of the skeleton as a whole changes periodically to provide accelerations measured on all bodies, from which the estimator can infer the bodies' relative orientations. Approximately every minute, the skeleton is moved along the world-x-axis with the position profile displayed in the top plot. The peaks in the bottom plot are the motion of the top plot repeated 75 times.

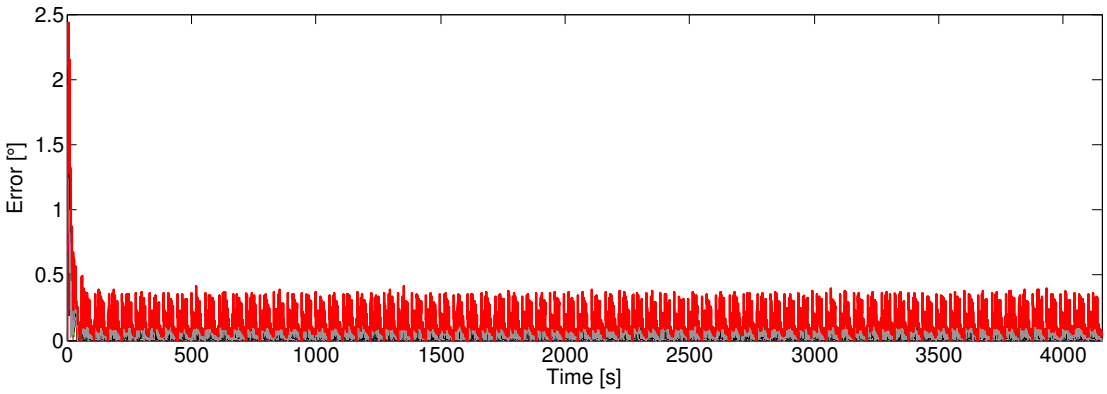


Figure 5.13 *Long Term Simulation Inclination Error*
The inclination errors (thin lines) of the same experiment as in Fig. 5.11 stay very small. On a very small scale, the movement of the skeleton as a whole also has an effect on the inclination error. The inclination errors, ϕ_k, are computed using eq. (5.14). While the individual errors are barely visible, the thick red line traces the maximum of the inclination errors, i. e. $\max(\phi_k \mid 1 \leq k \leq N)$, with the number of bodies N.

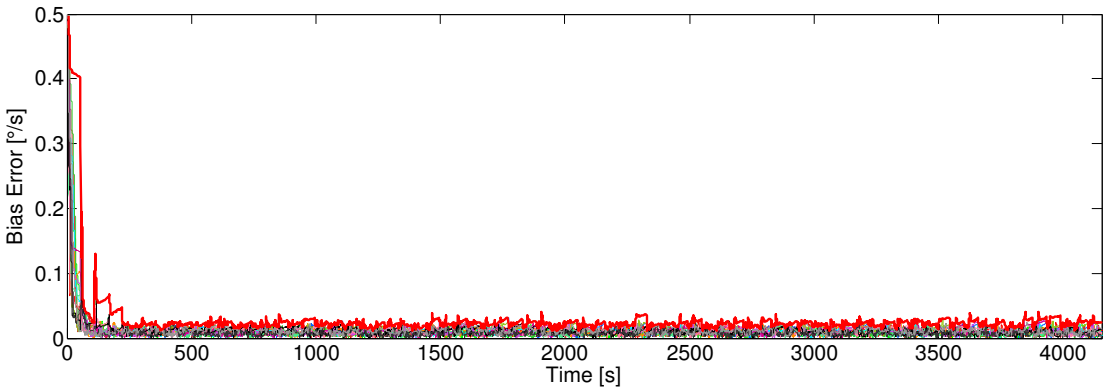


Figure 5.14 *Sirka Simulation: Gyroscope Bias Errors*
For a point in time, $b_{(k)}$ is the estimated gyroscope bias of body k in the estimator state, and $b_{(k)}^{(\mathrm{gt})}$ the corresponding ground truth bias from the simulation. Plotted in thin lines are the norms of the individual gyroscope bias errors, $\left\| b_{(k)} - b_{(k)}^{(\mathrm{gt})} \right\|_2$. The thick red line traces the maximum bias error, i. e. $\max\left\{ \left\| b_{(k)} - b_{(k)}^{(\mathrm{gt})} \right\|_2 \middle| \, 1 \leq k \leq N \right\}$, with the number of bodies N. Errors in the estimated gyroscope biases drop quickly.

5.10 Real World: Posture Estimation on a Shipyard

The most important experiment is, if course, to test the suit in the targeted environment. To prepare the test, I let one of the Meyer Werft's workers wear the sensor suit and do the calibration exercise. With the suit calibrated, the worker performed various tasks, which were both recorded by the suit and a video camera.

The tasks include kneeling down, lifting a cylinder and also welding in a steel cube, which would be impossible to track using a suit employing magnetometers.

No ground truth orientations to compare the estimated orientations to are available for the real-world experiments. To judge the estimator's performance, Fig. 5.15 and Fig. 5.16 juxtapose renderings of its estimate to videos of the worker performing the tasks.

Because the heading of the estimated posture as a whole is arbitrary, the rendering as a whole has been rotated around the vertical axis to approximately match the heading of the worker.

The application of the estimator is to provide the data basis to classify postures into categories like kneeling, kneeling on one knee only, working overhead, and so on. Additionally, the configuration of the bodies is supposed to be judged by a physiotherapy expert.

Thus, the postures should be recognizable in the renderings and should also approximately look like the postures of the worker in the video.

The rendering itself consists of one pyramid per tracked body. The origin of the physical coordinate system of a body lies in the base of the corresponding pyramid, whose apex points in the direction of the body's physical z-axis.

Instructive frames of the videos with renderings juxtaposed are reproduced in Figs. 5.15 and 5.16 for the lifting-and-kneeling[3] and welding[4] tasks, respectively.

For the lifting task, the rendering includes the result of an overly simplistic bend-over detection. Evaluating estimated postures is technically beyond the scope of this work, but it is both easy to make and a nice demonstration of the second intended application of the suit, namely to warn about potentially harmful postures.

In this case, it is deemed potentially harmful if the second top-most body of the torso, i. e. body 12 in the skeleton structure in Fig. 5.1, exceeds an inclination threshold θ_{harmful}:

$$\left| \angle \left(Q_{W\curvearrowleft(12)}^{T} \, [0 \;\; 0 \;\; 1]^T \, , [0 \;\; 0 \;\; 1]^T \right) \right| > \theta_{\text{harmful}} \tag{5.15}$$

As pictured in Fig. 5.15, the posture of the suit is apparently accurately tracked. The left column of Fig. 5.15 shows the test person lift up and put down a metal cylinder. When reaching the ground, the posture is judged to be potentially harmful by the simple criterion in eq. (5.15). In the right column, the subject lifts the cylinder by crouching and thus avoids the warning.

The lifting experiment in the right column of Fig. 5.15 exposes a principal problem of the suit: The sensors are not rigidly attached to the suit's wearer. In the bottom right image, the left-

[3] *http://www.informatik.uni-bremen.de/agebv2/downloads/videos/wenk_diss/owasexercise.mp4*
[4] *http://www.informatik.uni-bremen.de/agebv2/downloads/videos/wenk_diss/welding.mp4*

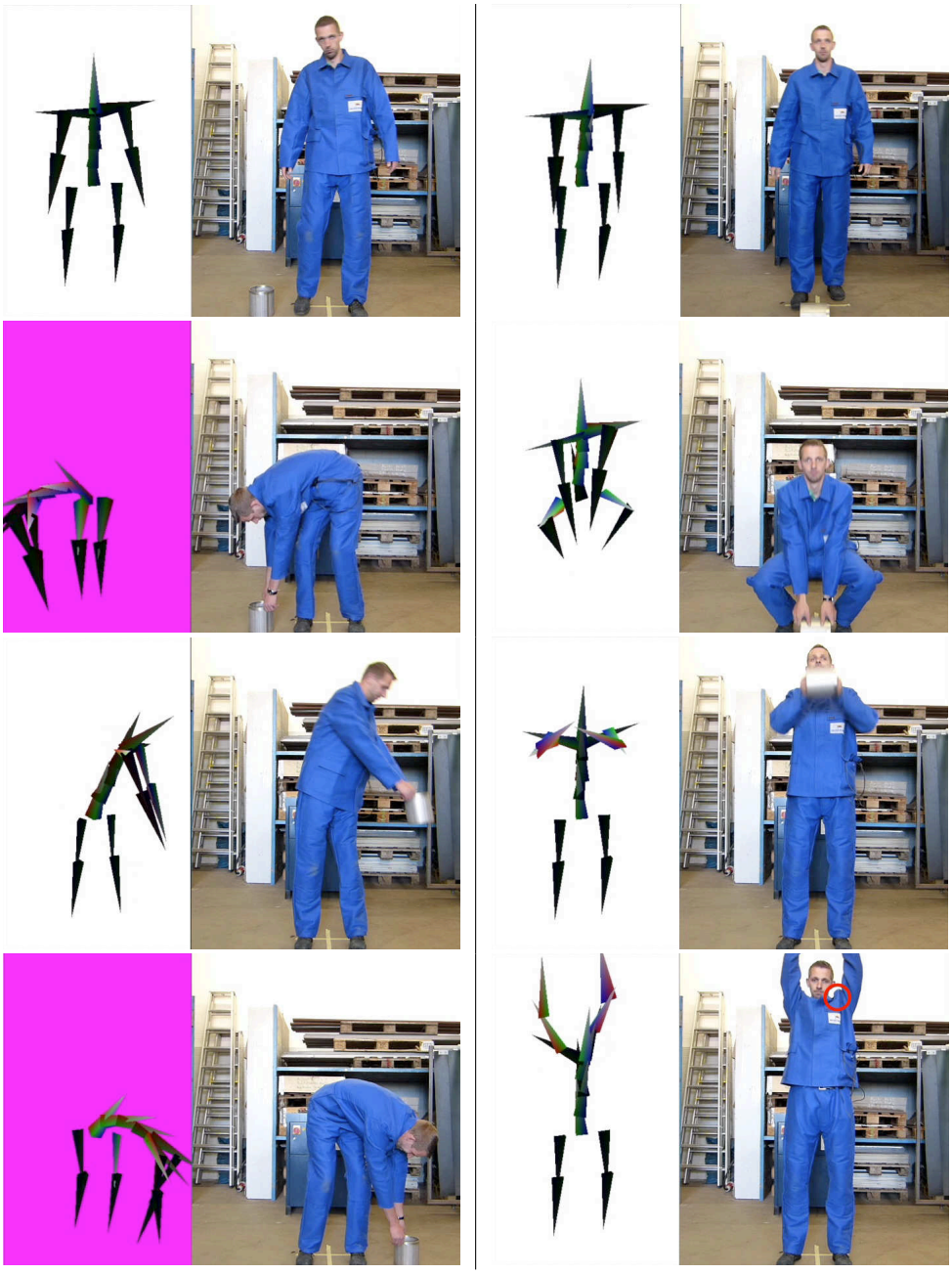


Figure 5.15 *Real-World Movements*
A worker equipped with the SIRKA suit lifts a heavy metal cylinder two different ways. The rendering's background becoming pink signals bend-over detection. In the left column, the cylinder is lifted and put down by bending over, in the right column the cylinder is picked up and lifted by crouching.

Figure 5.16 *Real-World Welding Experiment*
A worker equipped with the SIRKA suit welds in three different postures. The rendering consists of a pyramid for each body, from physical origin in physical z-direction.

neck-body, i. e. the body between spine and shoulder, is oriented almost perpendicularly to the right-neck-body. However, the subject's posture appears to be almost symmetric.

A closer look on the suit reveals a fold in the clothing approximately where the sensor of the body with the wrong orientation is mounted. The location is marked by a red circle. What happened is that the entire hook-and-loop-tape, in which the sensor is covered, rotated. This happens when the suit is strongly deformed, as the region around the collar is when lifting the arms overhead, and has to be dealt with when classifying postures. Despite the wrong orientation of the left-neck-body, it is still clear from the posture estimates that both arms are overhead.

The main event of the experiment was, of course, estimating postures of a worker while welding, not least because this would not be possible with any of the existing suits using magnetometers.

Frames of the video and corresponding posture estimate are shown in Fig. 5.16. The frames are ordered row-wise. The different postures while welding are easily recognized in the posture estimate. Although it is subtle, a slight inaccuracy of the orientation of the left lower arm is visible in the bottom-left image. Putting on "accessories", such as gloves, deforms the suit a little, which might lead to small orientation errors.

However, the estimates, obtained in real-time, are clearly accurate enough to distinguish postures of a welding worker.

Welding might also induce disturbances in the communications between sensor nodes and the central sensor fusion computer. Over the entire experiment, only four expected accumulates did not arrive at the sensor fusion computer. All at the same time, sensors of bodies 8, 9, 13 and 14 failed to respond. These are the sensors on the blue bus in Fig. 5.4. What probably happened is that the broadcast to request data got corrupted on that bus, but only once.

Such disturbances are unproblematic because of a nice side effect of the accumulation of IMU data to decouple sensor sampling and estimation rate as described in Sect. 4.7. If an accumulate does not arrive at the sensor fusion computer, data is not completely lost; only the effective accumulation duration gets longer, slightly degrading the quality of the data.

5.11 Recovering from Large Orientation Errors

The estimator appears to follow the actual posture quite accurately. Nevertheless, it is interesting to see how the estimator recovers from large orientation errors.

To test this, the estimator was fed with the accumulated inertial sensor data also used to test the calibration of a real suit in Sect. 5.8. The orientation estimator was initialized with the orientations from the beginning of the calibration exercise as found by the calibrator. Thus, using the same data, the orientation estimator started off with a very good initial estimate.

The estimator was run a second time with a large error injected in the orientation initialization: The orientation of body 2, that is the lowest body of the torso, was disturbed by 90° around the world-vertical axis. Since the rotation is around the direction of gravity, the estimator was

Figure 5.17 *Recovery from Large Errors*
While standing, a 90° error has been injected into the orientation of the bottom-most body of the torso, making the legs appear to be one-after-another instead of side-by-side (top-left). The estimator recovers while the worker walks for about 10s. The frames are ordered left to right, i. e. the last image is on the bottom-right.

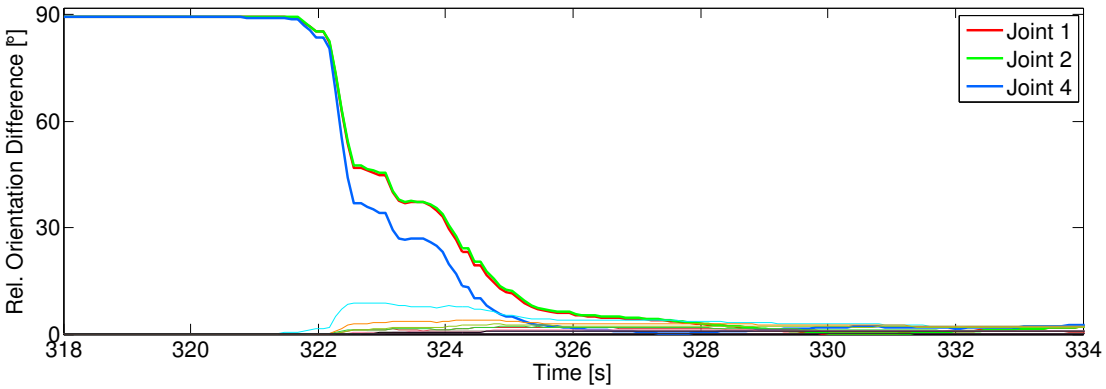


Figure 5.18 *Recovery from Large Errors: Orientation Differences*
Norms of the differences between the relative orientation estimates from an estimator with and an estimator without an injected error. Since the error is injected into the orientation of body 2, the interesting relative orientations are over joints 1, 2 and 4. The corresponding graphs are plotted thick.

not expected to recover from the error, as long as no accelerations in a direction different from gravity occurred. However, when the wearer of the suit started moving, the error should have started improve.

This is indeed what happened. It can be watched in another video[5], again with the rendering of the estimate and the actual motion juxtaposed. With the lowest part of the torso rotated by a right angle, the legs appear one-behind-another in the beginning of the experiment. At the end of the experiment, the estimator recovered from the error, such that the legs are again side-by-side, as they should be. The most instructive frames of the video are reproduced in Fig. 5.17.

The norms of the differences between the estimated relative orientations of the two estimation runs, first without and second with the injected error, are plotted in Fig. 5.18. The relative orientations over the joints 1, 2 and 4 are of particular interest, because they all connect body 2, whose initial orientation was subjected to the error.

The differences of the relative orientations did not change from the injected error until the wearer of the suit started moving, just as expected. In the first 6 seconds of movement, however, the differences vanished completely, indicating that the estimator recovered from the error.

[5] *http://www.informatik.uni-bremen.de/agebv2/downloads/videos/wenk_diss/error-recovery-annotated.mp4*

Chapter 6

Conclusion

Chapter 3 explored different priors to assume when orientations are to be estimated. The first assumption, that the accelerometer measures negative gravity on long term average, which is equivalent to the actual acceleration being zero on long term average, was known to work from the state of the art. That the other two, the velocity being zero or the position being zero on long term average, also work, suggests that it is worth trying other assumptions on other, maybe more indirect quantities, which the estimator can correlate to the orientation.

That there are alternatives to comparing the (possibly low-pass filtered) acceleration to gravity when the inclination is to be corrected is something that one should have at the back of ones mind when designing orientation estimators.

One example is the posture estimator of Chap. 4, where it was more convenient to use prior information on the velocity to correct the inclination, because it avoided numerically differentiating the noisy gyroscope signal and did not add any additional DOF to the state.

The posture estimator in Chap. 4 is very generic. It allows all joints of the skeleton to be spherical with an optional further restriction of a joint being a hinge. The estimator needs very few sensors, notably no magnetometers. Even in the absence of magnetometers, it recovers all relative orientations in all three degrees of freedom of jointed bodies as well as their globally accurate inclination. Using the accumulation of inertial sensor data to decouple the estimation rate from the sensor's sampling rate, it estimates postures on low-cost, embedded hardware in real-time with reasonable accuracy, although it estimates all body orientations at once. As a bonus, decoupling estimation rate and sensor sampling rate also adds robustness against data transmission failures.

The motion capturing workwear built using the estimator is easy to assemble, because it does not require the positions of the sensors in the workwear to be adjusted. They are calibrated instead, using (almost) the same models the posture estimator uses.

The end result is a very practical suit. It is straightforward to build and it can be used in motion-capturing-unfriendly environments: not just research laboratories, but real workplaces.

At the time of writing, SIRKA still is an ongoing project. The physiotherapists working on SIRKA are currently exploring what they can tell about the motions captured by the suit, beyond classification into standing, sitting, kneeling, bending over, etc. In part, this will determine how

useful motion capturing at the workplace really is. Presumably, there are a lot of postures which are fine if the worker has to carry only his own weight, but highly problematic if the worker carries some big mass. Measuring forces in addition to motion capturing appears to suggest itself for future work. That will not be easy. There are endless possibilities for a worker to make contact with his environment and to carry loads; just putting pressure sensors into the worker's shoes will not be enough.

Whether or not the SIRKA suit with my sensor fusion will be used for that future work remains to be seen. There is still quite some gap to bridge to make SIRKA a product ready to be sold. Noitom and Xsens already did that for their suits. In the industrial niche, however, there has been so much commercial interest in SIRKA that I believe the SIRKA suit is a serious candidate.

Appendix A

Posture From Motion Jacobians

The estimator of all orientations is implemented as an Extended Kalman Filter. The EKF needs the derivatives with respect to the state and inputs of the dynamics and (pseudo-)measurement model functions.

I chose the EKF over the Unscented Kalman Filter mainly because it requires only a single evaluation of the models per iteration instead of one evaluation per sigma point and the computation of the sigma points in the first place. Numerical computation of the Jacobian matrices would thus defeat the primary advantage of the EKF.

Instead, I derived the Jacobians analytically, which makes them very cheap to compute. Even better, computing the analytically derived Jacobians includes many terms which appear in the calculation of the model functions themselves.

To derive the Jacobians, the concrete definitions of $\boxplus$ and $\boxminus$ for rotation matrices are used. Thus, the following Jacobians are only valid for the operators defined in eq. (2.47).

State Dynamics Model Jacobian Matrices

Derivative with respect to State: $\frac{\partial}{\partial X} f(X, u)$ The dynamics model function is given in eq. (4.48):

$$f(X_{i-1}, u_i) = \left[f'(X_{i-1}^{(k)}, u_i^{(k)}) \right]_{k=1}^{N} \tag{A.1}$$

This is the function for accumulated sensor data, so for each body k,

$$u_i^{(k)} \equiv M_{j-n \leftarrow j}^{(k)} = \begin{bmatrix} Q_{j-n \leftarrow j}^{(k)} & v_{j-n \leftarrow j}^{(k)} \\ 0 & 1 \end{bmatrix} \tag{A.2}$$

is the relative orientation and velocity over one accumulation period. The accumulation period is worth one estimator iteration from $i-1$ to i and n raw sensor data samples from $j-n$ to j.

The dynamics of one body does not depend on the dynamics of a different body, so the Jacobian has a block-diagonal structure. Dropping all estimator-iteration indices for simplicity, the

Jacobian is

$$\frac{\partial}{\partial X} f(X,u) = \text{blkdiag}\left(\left[\frac{\partial}{\partial X^{(k)}} f'\left(X^{(k)}, u^{(k)}\right)\right]_{k=1}^{N}\right) \quad , \tag{A.3}$$

where f' is the dynamics model function for an individual body from eq. (4.46), repeated here again with estimator-iteration indices dropped:

$$f'(X^{(k)}, u^{(k)}) = \begin{bmatrix} Q_{W\curlywedge(k)} Q^{(k)}_{j-n\curlywedge j} \operatorname{Rot}(-b^{(k)}\delta T) \\ v^{(k)} + Q_{W\curlywedge(k)} \tilde{v}^{(k)}_{j-n\curlywedge j} + g\delta T \\ b^{(k)} \end{bmatrix} \tag{A.4}$$

$$\text{with } \tilde{v}^{(k)}_{j-n\curlywedge j} = v^{(k)}_{j-n\curlywedge j} - \frac{\delta T}{2} b^{(k)} \times v^{(k)}_{j-n\curlywedge j} \quad .$$

Since the portion of the state for body k contains its orientation, velocity and gyroscope bias, the portion of the Jacobian for body k contains the derivatives with respect to those components:

$$\frac{\partial}{\partial X^{(k)}} f'\left(X^{(k)}, u^{(k)}\right) = \left[\frac{\partial}{\partial Q_{W\curlywedge(k)}} f'\left(X^{(k)}, u^{(k)}\right) \quad \frac{\partial}{\partial v^{(k)}} f'\left(X^{(k)}, u^{(k)}\right) \quad \frac{\partial}{\partial b^{(k)}} f'\left(X^{(k)}, u^{(k)}\right)\right] \tag{A.5}$$

The simplest derivative is with respect to the velocity because only the velocity component depends on it:

$$\frac{\partial}{\partial v^{(k)}} f'\left(X^{(k)}, u^{(k)}\right) = \begin{bmatrix} 0 & I & 0 \end{bmatrix}^T \tag{A.6}$$

Deriving with respect to the orientation $Q_{W\curlywedge(k)}$ is a little tricker. Because $Q_{W\curlywedge(k)}$ is modified by applying small vectorially represented orientation changes using $\boxplus$, the Jacobian contains the vectorially represented change of the function with respect to a vectorially represented change at $q = 0$ to $Q_{W\curlywedge(k)}$. So

$$\frac{\partial}{\partial Q_{W\curlywedge(k)}} f'\left(X^{(k)}, u^{(k)}\right) \equiv \frac{\partial}{\partial q} f'\left(X^{(k)} \boxplus \begin{bmatrix} q \\ 0 \\ 0 \end{bmatrix}, u^{(k)}\right) \boxminus f'(X^{(k)}, u^{(k)})\Big|_{q=0} \tag{A.7}$$

$$= \frac{\partial}{\partial q} \begin{bmatrix} (Q_{W\curlywedge(k)} \boxplus q) Q^{(k)}_{j-n\curlywedge j} \operatorname{Rot}(-b^{(k)}\delta T) \boxminus Q_{W\curlywedge(k)} Q^{(k)}_{j-n\curlywedge j} \operatorname{Rot}(-b^{(k)}\delta T) \\ v^{(k)} + (Q_{W\curlywedge(k)} \boxplus q)\tilde{v}^{(k)}_{j-n\curlywedge j} + g\delta T - (v^{(k)} + Q_{W\curlywedge(k)} \tilde{v}^{(k)}_{j-n\curlywedge j} + g\delta T) \\ b^{(k)} - b^{(k)} \end{bmatrix}\Bigg|_{q=0} \quad . \tag{A.8}$$

Using the definitions of $\boxplus$, $\boxminus$ for orientations, Rot and aRot from eq. (2.47), and naming $R =$

$Q^{(k)}_{j-n\curvearrowleft j}\,\mathrm{Rot}(-b^{(k)}\delta T)$, this greatly simplifies to

$$\frac{\partial}{\partial q} f' \Big|_{q=0} = \frac{\partial}{\partial q} \begin{bmatrix} \mathrm{aRot}\left(R^T \exp(q_\times) R\right) \\ Q_{W\curvearrowleft(k)} \exp(q_\times) \tilde{v}^{(k)}_{j-n\curvearrowleft j} \\ 0 \end{bmatrix} \Bigg|_{q=0} = \frac{\partial}{\partial q} \begin{bmatrix} \mathrm{aRot}(\mathrm{Rot}(R^T q)) \\ Q_{W\curvearrowleft(k)} \exp(q_\times) \tilde{v}^{(k)}_{j-n\curvearrowleft j} \\ 0 \end{bmatrix} \Bigg|_{q=0} \tag{A.9}$$

$$= \begin{bmatrix} \left(Q^{(k)}_{j-n\curvearrowleft j}\,\mathrm{Rot}(-b^{(k)}\delta T)\right)^T \\ -Q_{W\curvearrowleft(k)} \left[\tilde{v}^{(k)}_{j-n\curvearrowleft j}\right]_\times \\ 0 \end{bmatrix} . \tag{A.10}$$

The orientation component of eq. (A.10) follows from using definitions and properties of the cross-product and matrix exponential map. The bias component is obvious.

The velocity component in the middle is a little bit tricky. To get there, one needs to exploit that the function is differentiated at $q = [q_x \;\; q_y \;\; q_z]^T = 0$. Dropping all indices for simplicity,

$$\frac{\partial}{\partial q} Q \exp(q_\times) v \Big|_{q=0} = Q \; \frac{\partial}{\partial q} \exp(q_\times) v \Big|_{q=0} \tag{A.11}$$

$$= Q \left(\frac{\partial}{\partial q_x} \exp(q_\times) v \Big|_{q_x=0} + \frac{\partial}{\partial q_y} \exp(q_\times) v \Big|_{q_y=0} + \frac{\partial}{\partial q_z} \exp(q_\times) v \Big|_{q_z=0} \right) . \tag{A.12}$$

Since

$$\exp(q_\times) = \exp\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_\times q_x + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_\times q_y + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_\times q_z \right) , \tag{A.13}$$

this yields

$$\frac{\partial}{\partial q} Q \exp(q_\times) v \Big|_{q=0} = Q \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}_\times v + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}_\times v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}_\times v \right) = Q v_\times^T = -Q v_\times \; . \tag{A.14}$$

Since the gyroscope bias is very small – of the anyway small bias the estimator only tracks the deviations not accounted for when accumulating sensor data on the sensor boards – I differentiate

the dynamics function with respect to the bias at $b = 0$.

$$\frac{\partial}{\partial b^{(k)}} f'(X^{(k)}, u^{(k)}) \approx \frac{\partial}{\partial b^{(k)}} f' \left(\begin{bmatrix} Q^{(k)} \\ v^{(k)} \\ 0 + b^{(k)} \end{bmatrix}, u^{(k)} \right) \boxminus f' \left(\begin{bmatrix} Q^{(k)} \\ v^{(k)} \\ 0 \end{bmatrix}, u^{(k)} \right) \Bigg|_{b^{(k)}=0} \tag{A.15}$$

$$= \frac{\partial}{\partial b^{(k)}} \begin{bmatrix} Q_{W\urcorner(k)} Q^{(k)}_{j-n\urcorner j} \operatorname{Rot}(-b^{(k)}\delta T) \boxminus Q_{W\urcorner(k)} Q^{(k)}_{j-n\urcorner j} \\ -Q_{W\urcorner(k)} \frac{\delta T}{2} b^{(k)} \times v_{j-n\urcorner j^{(k)}} \\ b^{(k)} \end{bmatrix} \Bigg|_{b^{(k)}=0} \tag{A.16}$$

$$= \begin{bmatrix} -I\delta T \\ Q_{W\urcorner(k)} \frac{\delta T}{2} [v_{j-n\urcorner j}]_\times \\ I \end{bmatrix} . \tag{A.17}$$

This completes the components of the derivative of the state dynamics function with respect to the state.

Derivative with respect to Accumulates: $\frac{\partial}{\partial u} f(X, u)$ These are derived analogously to the derivatives with respect to the state. The block structure is the same:

$$\frac{\partial}{\partial u} f(X, u) = \operatorname{blkdiag} \left(\left[\frac{\partial}{\partial u^{(k)}} f' \left(X^{(k)}, u^{(k)} \right) \right]_{k=1}^{N} \right) . \tag{A.18}$$

Each block is the derivative with respect to the accumulated sensor data of the corresponding sensor.

$$\frac{\partial}{\partial u^{(k)}} f' \left(X^{(k)}, u^{(k)} \right) = \left[\frac{\partial}{\partial Q^{(k)}_{j-n\urcorner j}} f' \left(X^{(k)}, u^{(k)} \right) \quad \frac{\partial}{\partial v^{(k)}_{j-n\urcorner j}} f' \left(X^{(k)}, u^{(k)} \right) \right] \tag{A.19}$$

The accumulated orientation is represented by a rotation matrix, so the derivative is again with respect to the orientation's 3-dimensional parameterization:

$$\frac{\partial}{\partial Q^{(k)}_{j-n\urcorner j}} f' \left(X^{(k)}, u^{(k)} \right) \equiv \frac{\partial}{\partial q} f' \left(X^{(k)}, u^{(k)} \boxplus \begin{bmatrix} q \\ 0 \end{bmatrix} \right) \boxminus f'(X^{(k)}, u^{(k)}) \Bigg|_{q=0} \tag{A.20}$$

$$= \frac{\partial}{\partial q} \begin{bmatrix} Q_{W\urcorner(k)} (Q^{(k)}_{j-n\urcorner j} \boxplus q) \operatorname{Rot}(-b^{(k)}\delta T) \boxminus Q_{W\urcorner(k)} Q^{(k)}_{j-n\urcorner j} \operatorname{Rot}(-b^{(k)}\delta T) \\ v^{(k)} + Q_{W\urcorner(k)} \tilde{v}^{(k)}_{j-n\urcorner j} + g\delta T - (v^{(k)} + Q_{W\urcorner(k)} \tilde{v}^{(k)}_{j-n\urcorner j} + g\delta T) \\ b^{(k)} - b^{(k)} \end{bmatrix} \Bigg|_{q=0} \tag{A.21}$$

$$= \begin{bmatrix} \operatorname{Rot}(-b^{(k)}\delta T) \\ 0 \\ 0 \end{bmatrix} . \tag{A.22}$$

The derivative with respect to the accumulated velocity is

$$\frac{\partial}{\partial v^{(k)}_{j-n\curvearrowleft j}} f'(X^{(k)}, u^{(k)}) = \begin{bmatrix} 0 \\ Q_{W\curvearrowleft(k)} \left(\frac{\partial}{\partial v^{(k)}_{j-n\curvearrowleft j}} v^{(k)}_{j-n\curvearrowleft j} - \frac{\delta T}{2} b^{(k)} \times v^{(k)}_{j-n\curvearrowleft j} \right) \\ 0 \end{bmatrix} \tag{A.23}$$

$$= \begin{bmatrix} 0 \\ Q_{W\curvearrowleft(k)} \left(I - \frac{\delta T}{2} b^{(k)}_{\times} \right) \\ 0 \end{bmatrix} \quad . \tag{A.24}$$

Measurement/Prior Model Jacobian Matrices

Derivative with respect to State: $\frac{\partial}{\partial X} h(X, \Omega)$ The measurement model function defined in eq. (4.19) is

$$0 = h(X, \Omega) + \zeta, \quad \zeta \sim \mathcal{N}(0, \Sigma_\zeta) \tag{A.25}$$

$$h(X, \Omega) = \begin{bmatrix} v^T_{1\dots N} & h_j(X, \Omega) & h_s(X) & [0 \;\; 1 \;\; 0] Q_{W\curvearrowleft(1)} [1 \;\; 0 \;\; 0]^T \end{bmatrix}^T \quad . \tag{A.26}$$

Its Jacobian does not have a block matrix structure as nice as the Jacobians of the dynamics function, because the positions of the entires of the joint constraint depend on the skeleton structure.

However, the general structure of the measurement function is, of course, reflected in its Jacobian.

$$\frac{\partial}{\partial X} h(H, \Omega) = \begin{bmatrix} \frac{\partial}{\partial X} [v_{1\dots N}] \\ \frac{\partial}{\partial X} h_j(X, \Omega) \\ \frac{\partial}{\partial X} h_s(X) \\ \frac{\partial}{\partial X} [0 \;\; 1 \;\; 0] Q_{W\curvearrowleft(1)} [1 \;\; 0 \;\; 0]^T \end{bmatrix} \tag{A.27}$$

$\frac{\partial}{\partial X} [v_{1\dots N}]$ is easy; it is zero everywhere except for the (3×3) blocks corresponding to the velocity components in the state. Since the state, defined in eq. (4.7), has the velocity as the second component of each body's state,

$$\frac{\partial}{\partial X} [v_{1\dots N}] = \text{blkdiag}(\overbrace{[0 \;\; I \;\; 0]}^{N\text{times}}) \quad . \tag{A.28}$$

The pseudo-measurement for the joint constraint of joint k is given in eq. (4.25). The individual constraints are stacked to form $h_j(X, \Omega)$. So for M joints and N sensors, $H_{j,X} = \frac{\partial}{\partial X} h_j(X, \Omega) \in \mathbb{R}^{3M \times 9N}$.

Let $[H_{j,X}]_{k,l} \in \mathbb{R}^{3\times 3}$ be the k'th and l'th (3×3) block matrix. These blocks are all zero except for the blocks of joint constraint k and the state components corresponding to the states of the bodies $\lambda(k)$ and $\mu(k)$, which joint constraint k depends on.

For all joints $1 \leq k \leq M$:

$$[H_{j,X}]_{k,\lambda(k)} = \frac{\partial}{\partial Q_{W\lambda(k)}} J_k = Q_{W\lambda(k)}[\psi^{(\lambda(k))}]_\times \tag{A.29}$$

$$[H_{j,X}]_{k,\mu(k)} = \frac{\partial}{\partial Q_{W\mu(k)}} J_k = -Q_{W\mu(k)}[\psi^{(\mu(k))}]_\times \tag{A.30}$$

$$[H_{j,X}]_{k,\lambda(k)+1} = \frac{\partial}{\partial v^{(\lambda(k))}} J_k = -I \tag{A.31}$$

$$[H_{j,X}]_{k,\mu(k)+1} = \frac{\partial}{\partial v^{(\mu(k))}} J_k = I \tag{A.32}$$

$$[H_{j,X}]_{k,\lambda(k)+2} = \frac{\partial}{\partial b^{(\lambda(k))}} J_k = -Q_{W\lambda(k)}[r_k^{(\lambda(k))}]_\times \tag{A.33}$$

$$[H_{j,X}]_{k,\mu(k)+2} = \frac{\partial}{\partial b^{(\mu(k))}} J_k = Q_{W\mu(k)}[r_k^{(\mu(k))}]_\times \tag{A.34}$$

Derivatives by orientations are, as they are for the dynamics function, with respect to the $\boxplus$-parameterization at zero.

The derivatives of the hinge constraints, given in eq. (4.32), are structured analogously. $H_s = \frac{\partial}{\partial X} h_s(X) \in \mathbb{R}^{3L \times 9N}$ for L hinges and N sensors. The (3×3) blocks of H_s are all zero except the following. For all $1 \leq k \leq L$

$$[H_s]_{k,\lambda(k)} = \frac{\partial}{\partial Q_{W\lambda(k)}} h_{s,k}(X) = Q_{W\lambda(k)}[a_k^{(\lambda(k))}]_\times \tag{A.35}$$

$$[H_s]_{k,\mu(k)} = \frac{\partial}{\partial Q_{W\mu(k)}} h_{s,k}(X) = -Q_{W\mu(k)}[a_k^{(\mu(k))}]_\times \ , \tag{A.36}$$

again differentiated with respect to the $\boxplus$-parameterization at zero.

The last row of the Jacobian $\frac{\partial}{\partial X} h(X, \Omega)$ corresponds to the heading constraint. Since the heading constraint is applied only to body 1, the constraint depends only on the orientation of body 1. Thus, all but the first three numbers of the last row of the Jacobian are 0:

$$\frac{\partial}{\partial X}[0\ 1\ 0] Q_{W\mathsf{1}}[1\ 0\ 0]^T = \left[\frac{\partial}{\partial Q_{W\mathsf{1}}}[0\ 1\ 0] Q_{W\mathsf{1}}[1\ 0\ 0]^T \ \ 0\right] \tag{A.37}$$

$$= \left[[0\ 1\ 0]\left(\frac{\partial}{\partial Q_{W\mathsf{1}}} Q_{W\mathsf{1}}[1\ 0\ 0]^T\right) \ \ 0\right] \tag{A.38}$$

$$= -\left[[0\ 1\ 0]\, Q_{W\mathsf{1}}\left[[1\ 0\ 0]^T\right]_\times \ \ 0\right] \tag{A.39}$$

$$= \left[\begin{bmatrix} 0 & -[Q_{W\mathsf{1}}]_{2,3} & [Q_{W\mathsf{1}}]_{2,2} \end{bmatrix} \ \ 0\right] \ . \tag{A.40}$$

Again, the derivative by the orientation matrix is with respect to its $\boxplus$-parameterization at 0.

Derivative with respect to angular velocities: $\frac{\partial}{\partial \Omega} h(X, \Omega)$ Only the joint constraint $h_j(X, \Omega)$ depends on the angular velocities. So $\frac{\partial}{\partial \Omega} h(X, \Omega)$ looks structurally like $\frac{\partial}{\partial X} h(X, \Omega)$ of eq. (A.27),

but is zero almost everywhere:

$$\frac{\partial}{\partial \Omega} h(H, \Omega) = \begin{bmatrix} 0 \\ \frac{\partial}{\partial \Omega} h_j(X, \Omega) \\ 0 \\ 0 \end{bmatrix} \tag{A.41}$$

For M joints and N sensors $H_{j,\Omega} = \frac{\partial}{\partial \Omega} h_j(X, \Omega) \in \mathbb{R}^{3M \times 3N}$. $H_{j,\Omega}$ is smaller than $H_{j,X}$, because there are 9 DOF per body state in X, but only 3 DOF per angular velocity in Ω. Let $[H_{j,\Omega}]_{k,l} \in \mathbb{R}^{3\times 3}$ be the k'th and l'th (3×3) block matrix.

The non-zero blocks are again derivatives of J_k, as they were when differentiating with respect to X in eqs. (A.29) to (A.34).

The derivative of J_k with respect to one of the connected bodies' angular velocity is the same as the derivative with respect to the same body's gyroscope bias, just with the sign flipped. For for all joints $1 \leq k \leq M$,

$$[H_{j,\Omega}]_{k,\lambda(k)+2} = \frac{\partial}{\partial \omega^{(\lambda(k))}} J_k = Q_{W\urcorner\lambda(k)} [r_k^{(\lambda(k))}]_\times \tag{A.42}$$

$$[H_{j,\Omega}]_{k,\mu(k)+2} = \frac{\partial}{\partial \omega^{(\mu(k))}} J_k = -Q_{W\urcorner\mu(k)} [r_k^{(\mu(k))}]_\times \tag{A.43}$$

This completes the Jacobian matrices used in the Extended Kalman Filter.

Appendix B

References

B.1 List of Figures

B.2 Bibliography

[ARD] ARD. *Wissen vor Acht: Anzug macht Haltung.* URL: *http://www.daserste.de/information/wissen-kultur/wissen-vor-acht-zukunft/sendung/wissen-vor-acht-zukunft-332.html*.

[ARM] ARM. *Cortex Microcontroller Software Interface Standard.* URL: *http://www.keil.com/pack/doc/CMSIS*.

[Bin74] Christopher Bingham. "An antipodally symmetric distribution on the sphere". In: *The Annals of Statistics* (1974), pp. 1201–1225.

[Bio] Ekso Bionics. *[Online].* URL: *https://www.eksobionics.com*.

[BKL02] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation.* New York, NY, USA: John Wiley & Sons, Inc., 2002. ISBN: 0471221279.

[Bos] Bosch. *BMI160 Data sheet.* URL: *https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMI160-DS000-07.pdf*.

[Bro+97] Ilja N. Bronstein, Konstantin A Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik.* Vol. 3. Verlag Harri Deutsch, 1997.

[Car+14] Nicola Carbonaro, Gabriele Dalle Mura, Federico Lorussi, Rita Paradiso, Danilo De Rossi, and Alessandro Tognetti. "Exploiting wearable goniometer technology for motion sensing gloves". In: *Biomedical and Health Informatics, IEEE Journal of* 18.6 (2014), pp. 1788–1795.

[CMC07] John L Crassidis, F Landis Markley, and Yang Cheng. "Survey of nonlinear attitude estimation methods". In: *Journal of Guidance, Control, and Dynamics* 30.1 (2007), pp. 12–28.

[CO10] Peng Cheng and Bengt Oelmann. "Joint-angle measurement using accelerometers and gyroscopes—A survey". In: *Instrumentation and Measurement, IEEE Transactions on* 59.2 (2010), pp. 404–414.

[EHS09] Rolf Ellegast, Ingo Hermanns, and Christoph Schiefer. "Workload Assessment in Field Using the Ambulatory CUELA System". In: *Digital Human Modeling*. Ed. by VincentG. Duffy. Vol. 5620. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 221–226. ISBN: 978-3-642-02808-3.

[Fav+09] Ja Favre, Rab Aissaoui, BMc Jolles, JAb de Guise, and K Aminian. "Functional calibration procedure for 3D knee joint angle description using inertial sensors". In: *Journal of biomechanics* 42.14 (2009), pp. 2330–2335.

[Fea08] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2008.

[Gil+16] Igor Gilitschenski, Gerhard Kurz, Simon J Julier, and Uwe D Hanebeck. "Unscented orientation estimation based on the Bingham distribution". In: *IEEE Transactions on Automatic Control* 61.1 (2016), pp. 172–177.

[GK13] Jared Glover and Leslie Pack Kaelbling. "Tracking 3-D rotations with the quaternion Bingham filter". In: (2013).

[Gro+04] Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. "Style-based inverse kinematics". In: *ACM transactions on graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 522–531.

[GV12] Gene H Golub and Charles F Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[Her+13] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. "Integrating Generic Sensor Fusion Algorithms with Sound State Representations through Encapsulation of Manifolds". In: *Information Fusion* 14.1 (2013), pp. 57–77. ISSN: 1566-2535.

[Her08] Christoph Hertzberg. "A Framework for Sparse, Non-Linear Least Squares Problems on Manifolds". Diploma Thesis. Universität Bremen, 2008.

[Ida96] M Idan. "Estimation of Rodrigues parameters from vector observations". In: *Aerospace and Electronic Systems, IEEE Transactions on* 32.2 (1996), pp. 578–586.

[II11] Vladimir G Ivancevic and Tijana T Ivancevic. "Lecture Notes in Lie Groups". In: *arXiv preprint arXiv:1104.1106* (2011).

[Ioz+12] LI Iozan, M Kirkko-Jaakkola, J Collin, J Takala, and C Rusu. "Using a MEMS gyroscope to measure the Earth's rotation for gyrocompassing applications". In: *Measurement Science and Technology* 23.2 (2012), p. 025005.

[JU97] Simon J. Julier and Jeffrey K. Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *Proc. SPIE*. Vol. 3068. 1997, pp. 182–193.

[Kal60] Rudolf E. Kalman. "A new approach to linear filtering and prediction problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82 (1960). Online: *http://www.cs.unc.edu/~welch/kalman/media/pdf/Kalman1960.pdf*, pp. 35–45.

[KHS14] Manon Kok, Jeroen Hol, and Thomas Schön. "An optimization-based approach to human body motion capture using inertial sensors". In: *19th World Congress of the*

International Federation of Automatic Control (IFAC), Cape Town, South Africa, August 24-29, 2014. International Federation of Automatic Control. 2014, pp. 79–85.

[Kok+12] Manon Kok, Jeroen D Hol, Thomas B Schön, Fredrik Gustafsson, and Henk Luinge. "Calibration of a magnetometer in combination with inertial sensors". In: *Information Fusion (FUSION), 2012 15th International Conference on*. IEEE. 2012, pp. 787–793.

[KR88] Brian W. Kernighan and Dennis M. Ritchie. *The C programming language, 2nd edition*. Upper Saddle River, New Jersey, USA: Prentice Hall, 1988. ISBN: 0-13-110362-8.

[Kra03] Edgar Kraft. "A quaternion-based unscented Kalman filter for orientation tracking". In: *Proceedings of the Sixth International Conference of Information Fusion*. Vol. 1. 2003, pp. 47–54.

[KTL07] Nives Klopčar, Martin Tomšič, and Jadran Lenarčič. "A kinematic model of the shoulder complex to evaluate the arm-reachable workspace". In: *Journal of biomechanics* 40.1 (2007), pp. 86–91.

[Liu+09] Kun Liu, Tao Liu, Kyoko Shibata, and Yoshio Inoue. "Ambulatory measurement and analysis of the lower limb 3D posture using wearable sensor system". In: *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*. IEEE. 2009, pp. 3065–3069.

[LMS82] Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. "Kalman filtering for spacecraft attitude estimation". In: *Journal of Guidance, Control, and Dynamics* 5.5 (1982), pp. 417–429.

[LRS11] H.J. Luinge, D. Roetenberg, and P.J. Slycke. *Inertial Sensor Kinematic Coupling*. US Patent App. 12/534,526. 2011. URL: *http://www.google.com/patents/US20110028865*.

[Lui02] H Luinge. "Inertial sensing of human motion". PhD thesis. PhD thesis. University of Twente, Netherlands, 2002.

[LV05] Henk J Luinge and Peter H Veltink. "Measuring orientation of human body segments using miniature gyroscopes and accelerometers". In: *Medical and Biological Engineering and computing* 43.2 (2005), pp. 273–282.

[LVB07] Henk J Luinge, Peter H Veltink, and Chris TM Baten. "Ambulatory measurement of arm orientation". In: *Journal of biomechanics* 40.1 (2007), pp. 78–85.

[Mar88] F Landis Markley. "Attitude determination using vector observations and the singular value decomposition". In: *The Journal of the Astronautical Sciences* 36.3 (1988), pp. 245–258.

[MB90] Gary Monheit and Norman I Badler. "A kinematic model of the human spine and torso". In: *Technical Reports (CIS)* (1990), p. 746.

[Mie+13] Markus Miezal, Gabriele Bleser, Norbert Schmitz, and Didier Stricker. "A generic approach to inertial tracking of arbitrary kinematic chains". In: *Proceedings of the 8th International Conference on Body Area Networks*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 2013, pp. 189–192.

[Noi] Noitom. *Perception Neuron*. URL: *https://neuronmocap.com/*.

[ODo+07] Karol J O'Donovan, Roman Kamnik, Derek T O'Keeffe, and Gerard M Lyons. "An inertial and magnetic sensor based technique for joint angle measurement". In: *Journal of biomechanics* 40.12 (2007), pp. 2604–2611.

[Psi00] Mark L Psiaki. "Attitude-determination filtering via extended quaternion estimation". In: *Journal of Guidance, Control, and Dynamics* 23.2 (2000), pp. 206–214.

[RDP16] Sam Royston, Connor DeFanti, and Ken Perlin. "A Collaborative Untethered Virtual Reality Environment for Interactive Social Network Visualization". In: *arXiv preprint arXiv:1604.08239* (2016).

[Rei+10] Attila Reiss, Gustaf Hendeby, Gabriele Bleser, and Didier Stricker. "Activity recognition using biomechanical model based pose estimation". In: *Smart Sensing and Context*. Springer, 2010, pp. 42–55.

[RLS09] Daniel Roetenberg, Henk Luinge, and Per Slycke. *Xsens MVN: full 6DOF human motion tracking using miniature inertial sensors*. Tech. rep. Xsens Motion Technologies BV, 2009.

[Roe+05] Daniel Roetenberg, Henk J Luinge, Chris Baten, and Peter H Veltink. "Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation". In: *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 13.3 (2005), pp. 395–405.

[Sal+13] Sarvenaz Salehi, Gabriele Bleser, Norbert Schmitz, and Didier Stricker. "A low-cost and light-weight motion tracking suit". In: *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. IEEE. 2013, pp. 474–479.

[Sal+15] Sarvenaz Salehi, Gabriele Bleser, Attila Reiss, and Didier Stricker. "Body-IMU Autocalibration for Inertial Hip and Knee Joint Tracking". In: BodyNets '15 (2015), pp. 51–57.

[Sim10] Dan Simon. "Kalman filtering with state constraints: a survey of linear and nonlinear algorithms". In: *IET Control Theory & Applications* 4.8 (2010), pp. 1303–1318.

[SRS14] Thomas Seel, Jörg Raisch, and Thomas Schauer. "IMU-based joint angle measurement for gait analysis". In: *Sensors* 14.4 (2014), pp. 6891–6909.

[SSR12] T. Seel, T. Schauer, and J. Raisch. "Joint axis and position estimation from inertial measurement data by exploiting kinematic constraints". In: *Control Applications (CCA), 2012 IEEE International Conference on*. Oct. 2012, pp. 45–49. DOI: *10.1109/CCA.2012.6402423*.

[Stu64] John Stuelpnagel. "On the parametrization of the three-dimensional rotation group". In: *SIAM review* 6.4 (1964), pp. 422–430.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[Tog+14] Alessandro Tognetti, Federico Lorussi, Gabriele Dalle Mura, Nicola Carbonaro, Maria Pacelli, Rita Paradiso, and Danilo De Rossi. "New generation of wearable goniometers for motion capture systems". In: *Journal of neuroengineering and rehabilitation* 11.1 (2014), p. 1.

[Tri] Trivisio. *Inertial Motion Tracking: Light motion suit*. URL: *http://media.wix.com/ugd/f221b8_a943273544d5421b9b8c98cf13835daf.pdf*.

[Wah65] Grace Wahba. "A least squares estimate of spacecraft attitude". In: *SIAM Review* 7.3 (1965), p. 409.

[WF15] Felix Wenk and Udo Frese. "Posture From Motion". In: *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015)*. Funded by the BMBF project SIRKA (16SV6238K). 2015, pp. 280–285.

[Won+15] C. Wong, Zhi-Qiang Zhang, B. Lo, and Guang-Zhong Yang. "Wearable Sensing for Solid Biomechanics: A Review". In: *Sensors Journal, IEEE* 15.5 (May 2015), pp. 2747–2760. ISSN: 1530-437X.

[You09] Alexander D Young. "Comparison of orientation filter algorithms for realtime wireless inertial posture tracking". In: *Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on*. IEEE. 2009, pp. 59–64.

[You10] A.D. Young. "Use of Body Model Constraints to Improve Accuracy of Inertial Motion Capture". In: *Body Sensor Networks (BSN), 2010 International Conference on*. June 2010, pp. 180–186. DOI: *10.1109/BSN.2010.30*.

[Zih+15] S. Zihajehzadeh, P.K. Yoon, Bong-Soo Kang, and E.J. Park. "UWB-Aided Inertial Motion Capture for Lower Body 3-D Dynamic Activity and Trajectory Tracking". In: *Instrumentation and Measurement, IEEE Transactions on* 64.12 (Dec. 2015), pp. 3577–3587. ISSN: 0018-9456. DOI: *10.1109/TIM.2015.2459532*.